UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA

Deep Learning for Cardiac MR Images Analysis

Cristiana Ferreira Tiago

Mestrado Integrado em Engenharia Biomédica e Biofísica
Perfil em Engenharia Clínica e Instrumentação Médica

Dissertação orientada por:
Prof. Dr. ir. Marcel Breeuwer
Prof. Dr. Alexandre Andrade

2018

Acknowledgments

I would like to start by expressing my deepest gratitude to everybody involved in this project.

In first place, I give a special appreciation to my supervisors at the Medical Image Analysis Group (IMAG/e) from the Eindhoven University of Technology (TU/e), Prof. Dr. Marcel Breeuwer and Dr. Mitko Veta. To Marcel for accepting me as his student, for trusting me this project, for helping me when I needed, for the casual conversations/meetings and much more, and to Mitko for all the support and advices given during these 10 months and for the social times.

Furthermore, Professor Alexandre Andrade, my supervisor back at the Faculty of Sciences of the University of Lisbon (FCUL), could not stay out this list. Thank you for taking care of everything during my stay in The Netherlands, for enlightening me about all the questions and for the advices given during this time until today. Also to Prof. Raquel Conceição, for the encouragement and revision of my texts.

I cannot forget the rest of the group with whom I spent a really good time while staying at TU/e: Koen, Maxime, Andrey, Friso, Pim, Samaneh, Veronika, Josien, Linde, Rina and Suzanne. Dank je wel! A special thanks goes to João and Leroy for all the time spent together having lunch or watching the World Cup matches, for helping me solving some of my problems, for staring with me at the asian guy rage and for the laughs. I'll take you both with me.

I would also like to say thank you to the best group of friends I gained back in Eindhoven: my roomies! Linh, Ilaria, Teodora and to my guys with whom I spent the last 6 months, Erwan, Jonathan and Mark. Thank you for all the international meals, conversations in the balcony, bike tours around Eindhoven, UNO nights and everything else. I had a blast with all of you! Inês, Marta, Natália, Rodrigo, Paulo e Nuno, obrigada pelas visitas e pelas conversas que tivemos. São vocês quem eu levo destes 5 anos.

A very special acknowledgment goes to my family. For trusting, supporting and hearing me while I was 1000 km away. Para os meus pais, por todos os sacrifícios que fizeram e por sempre acreditarem que eu conseguiria fazer qualquer coisa, mesmo quando eu não acreditava. Para o meu irmão, por me visitar, por ter feito um buraco na parede do meu quarto, por cozinhar a massa de frango para mim e especialmente pela companhia feita em Eindhoven e durante todos estes anos. Adoro-te chavalo!

Last but not least I want to thank to the Erasmus + program for the financial support given.

Abstract

In the current days cardiac functional parameters are measured from Computed Tomography (CT) scans or from echocardiographies using clinical softwares that rely on an experienced user to select relevant points and areas in the images. The used images are usually collected in the aorta or following a Short Axis (SA) cardiac plane. The parameters' quantification through Cardiac Magnetic Resonance (CMR) is also possible since this imaging modality provides better anatomical information about the left ventricle (LV) and when this is the chosen modality to obtain the heart images, the SA plane is, again, the most commonly used.

It would be useful to have a clinical software to quantify the more relevant cardiac functional parameters such as Stroke Volume (SV), Ejection Fraction (EF) and Cardiac Output (CO) with minimal user interaction from the image acquisition part to the final quantification of these parameters. Even though the most used plane to acquire images is the SA and the majority of the scientific results concern this image view, its usage presents a disadvantage when trying to quantify the LV volume. The SA images are acquired from the apex of the heart to the valve plane of the LV and then are used to extrapolate the LV volume. However, these 2 boundary positions are the most complicated to obtain and it affects the final value of the volume. To avoid this difficulty, in this project the considered plane was the Long Axis (LA) one, where one can see the apex and the valve points, and this plane was set in different areas of the heart producing three different views: the 2 Chamber Long Axis (2 CH-LA), the 3 Chamber Long Axis (3 CH-LA) and the 4 Chamber Long Axis (4 CH-LA) view.

This project aims to analyze the three different kinds of LA CMR images leading to the SV, EF and CO quantification while reducing the user interaction with the software. To achieve it Deep Learning (DL) methods, which belong to the Artificial Intelligence (AI) area and model the human brain behavior and function by creating artificial neurons as well as synapses in the form of Convolutional Neural Networks (CNNs), were developed and explored. These DL methods can perform classification tasks yielding results similar to the ones obtained by humans.

To quantify the parameters there is a quantity which is crucial to have, the LV volume. The followed methodology consisted in create CNNs and train them to perform the classification task of segmenting the LV in any LA image. The network training is realized by presenting a wide variety of labeled LA images, i.e. images where the area to segment, the LV, is already indicated, to the CNN and let it learn what to look for in a new image to correctly segment it. From the network LV area prediction it was possible to derivate the LV volume and from this quantify the SV, EF and CO.

The results of this methodology allow to analyze the CNNs' performance and the final parameters' values obtained from real patients' data in order to derive a conclusion about the DL potential to segment LA images and the creation of a user independent framework that could be translated into a clinical software.

Key Words: Convolutional Neural Network, Deep Learning, Left Ventricle, Long Axis.

Resumo

Atualmente os indicadores da função cardíaca, tais como o Volume Sistólico (VS), a Fração de Ejeção (FE) e o Débito Cardíaco (DC), são calculados a partir de exames de Tomografia Axial Computorizada (TAC) ou ecocardiografias usando *softwares* clínicos cuja utilização requer a experiência do utilizador que seleciona pontos e áreas da imagem relevantes para o cálculo final. As imagens usadas são, geralmente, obtidas na zona da aorta ou considerando o plano anatómico que segue o eixo curto (EC) cardíaco, i.e., o plano transversal, capturando apenas ambos os ventrículos. A quantificação destes indicadores através de imagens de Ressonância Magnética Cardíaca (RMC) também é possível, embora não seja tão usada devido ao elevado custo do exame por paciente, uma vez que esta modalidade fornece imagens com melhor informação anatómica sobre as estruturas cardíacas sendo o Ventrículo Esquerdo (VE) a mais importante uma vez que é a partir desta cavidade que o sangue flui para todo o corpo devido à ação contrativa do miocárdio. Mesmo utilizando a RMC como modalidade de imagem, o plano mais utilizado continua a ser o transversal.

Tendo em conta os progressos tecnológicos que hoje em dia se verificam, seria de grande utilidade o desenvolvimento de um *software* clínico para avaliar os parâmetros cardíacos acima referidos, entre outros, com a mínima interação do utilizador desde que se adquirem as imagens até que se faz o cálculo final dos valores dos indicadores. Sendo que o plano de imagem mais usado é o do EC e a maioria dos resultados provenientes de grupos de investigação nesta área consideram imagens com esta vista, ou seja, apenas com ambos os ventrículos. A obtenção de imagens segundo o EC passa por fazê-lo situando o plano em várias posições entre o ápice e o plano que inclui a válvula mitral, permitindo reconstruir o volume do VE. No entanto as imagens correspondentes ao plano da válvula e ao do ápice são complicadas de adquirir, o que afeta o valor final do volume. Para ultrapassar esta dificuldade, neste projeto foram utilizadas imagens cujo plano anatómico contém o eixo longo (EL) cardíaco. Nesta situação, existem 2 planos que permitem visualizar aurículas e ventrículos na mesma imagem, o plano longitudinal e o sagital. Consoante o escolhido, conseguem produzir-se 3 tipos de vistas diferentes, sempre seguindo o EL, onde se podem visualizar 2, 3 ou 4 cavidades cardíacas simultaneamente. Nas imagens das duas cavidades distingue-se o ventrículo e a aurícula esquerda, nas de três distinguem-se estas duas estruturas mais uma porção da aorta e nas de quatro é possível observar os dois ventrículos e as duas aurículas.

Este projeto tem como principal objetivo analisar imagens de RMC obtidas segundo o EL do coração levando à quantificação dos parâmetros VS, FE e DC reduzindo ao máximo a interação do utilizador com o *software*. Para tal, são usados métodos de DL.

No que toca ao desempenho de tarefas cognitivas, a melhor solução passa por utilizar o cérebro humano e todos os conhecimentos a ele associados. No entanto, por vezes, as tarefas cognitivas em questão são desafiadoras, complicadas e demoradas e nestas circunstâncias é benéfica a utilização de ferramentas que simulem o funcionamento do cérebro, deixando o utilizador livre para realizar outras funções ao mesmo tempo. Assim apareceu a Inteligência Artificial (IA) que permite modelar o comportamento e funções do cérebro humano através da criação de neurónios artificiais assim como das sinapses, ou seja, do comportamento fisiológico que explica a transmissão de informações entre neurónios, sob a forma de Redes Neuronais Convolucionais (RNCs). Estas RNCs permitem simular a inteligência humana assim como o processo de aprendizagem.

Durante os anos 80 e 90 apareceram as primeiras redes de neurónios artificiais, redes estas que não continham tantos neurónios como o cérebro humano mas que revelaram uma elevada capacidade para

resolver problemas de classificação, como por exemplo fazer a distinção entre uma imagem de um paciente doente e de um saudável, e regressão, criando uma nova área, a de *Machine Learning* (ML). Já durante os anos 2000 o DL apareceu, onde se podem encontrar redes com mais neurónios com capacidade para resolver problemas mais complexos e de maneira mais independente, como por exemplo segmentar vários órgãos numa só imagem, sendo muito utilizados nas mais variadas áreas do conhecimento incluindo o processamento de imagens médicas. Estes métodos de DL apresentam resultados muito próximos daqueles obtidos por especialistas.

A metodologia aqui usada passa por criar RNCs e treiná-las de modo a segmentarem o VE em qualquer tipo de imagem de RMC obtida segundo o EL do coração. O treino de uma rede neuronal passa por apresentar-lhe um elevado e variado número deste tipo de imagens onde o VE já se encontra identificado, isto é, já têm "legendas" e deixá-la procurar características que a própria rede considera mais importantes de modo a conseguir segmentar uma nova imagem, nunca antes vista durante a fase de treino. Neste projeto foram desenvolvidas várias RNCs, treinadas durante diferentes períodos de tempo e sujeitas a imagens com diferentes vistas e "legendas", prevendo diferentes estruturas.

Para atingir os objetivos deste projeto e quantificar os indicadores da função cardíaca é crucial saber o volume do VE que pode ser derivado a partir da área e do comprimento do EL do VE. Estas duas últimas variáveis são obtidas através dos resultados de segmentação das várias RNCs. Foram treinadas 8 redes diferentes: duas redes que segmentam o contorno do VE (U-Net_20000 e U-Net_50000), cinco que prevêem a área desta estrutura (U-Net_FilledMasks_20000, U-Net_FilledMasks_50000, U-Net_2CH, U-Net_3CH e U-Net_4CH) e uma que identifica 3 pontos chave numa imagem obtida segundo o EL (8^{th} *trained* U-Net). Ambas as redes que segmentam o contorno do VE, a que identifica os 3 pontos chave e duas das que prevêem a área do VE foram treinadas com recurso a um conjunto de imagens onde se viam 2, 3 ou 4 câmaras cardíacas, sendo que as restantes 3 foram treinadas usando conjuntos de imagens com vistas específicas. De modo a desenvolver o *software* que não dependa do utilizador para quantificar o VS, FE e DC, começou por usar-se o resultado da segmentação dos 3 pontos chave: o ápice do coração e os 2 pontos que definem o segmento de reta entre os limites da válvula mitral, obtendo as suas coordenadas de modo a medir o EL do ventrículo. De seguida, e usando as segmentações da área do VE em adição ao previamente calculado comprimento do eixo, foi calculado o volume do VE e, consequentemente, os indicadores da função cardíaca.

Novas imagens de RMC obtidas segundo o EL do coração foram utilizadas para avaliar tanto o potencial da utilização de métodos de DL na segmentação deste tipo de imagens como o desempenho das RNCs e deste *software* independente de um utilizador, com os resultados a mostrar que: (i) é mais complicado prever com exatidão pequenas áreas nas imagens do que grandes, daí as segmentações do contorno do VE não serem tão precisas quanto as da área, (ii) devido à conclusão anterior e ao facto dos resultados obtidos relativamente à segmentação das áreas do VE a partir de imagens obtidas segundo o EL estarem nivelados com os resultados considerados como estado-da-arte para esta tarefa, foram usadas as predições das áreas em detrimento dos contornos na criação do *software* independente do utilizador, (iii) até à data é-me desconhecida a existência de uma RNC que detete os 3 pontos chave aqui mencionados sendo que os resultados obtidos são satisfatórios e facilitam a criação do já mencionado *software*, (iv) os valores finais dos parâmetros cardíacos estão de acordo com os valores estabelecidos para referência e não dependem da proveniência da segmentação final da área do ventrículo esquerdo.

Concluindo, o objetivo inicial do projeto foi alcançado havendo espaço para futuras correções nomeadamente através da criação de "legendas" para as imagens mais exatas, de novas RNCs ou alteração das já existentes de modo as que as predições sejam semelhantes às produzidas por humanos ou utilização da mesma metodologia para analisar imagens de RMC obtidas segundo o EC do coração.

Devido ao tempo extra disponível, foi treinada uma nona RNC para, desta vez, resolver um problema de regressão. De modo a tentar avaliar qualitativamente o grau de oclusão das artérias coronárias, artérias estas que são responsáveis pela perfusão do miocárdio, foi utilizado o Modelo de *Tofts* (MT) para tentar prever o valor do parâmetro que representa a taxa de fluxo de sangue dos vasos para as células do tecido, neste caso do miocárdio a partir da evolução temporal de 2 sinais fisiológicos: da concentração do agente de contraste que entra no tecido e daquela que de facto se mede dentro deste. Treinando a rede neuronal com várias amostras geradas computacionalmente destes sinais, os resultados obtidos mostram que os valores previstos para o parâmetro mencionado não diferem muito dos originais, havendo, mesmo assim, margem de manobra para melhorar esta rede de regressão, e que a partir deste parâmetro é possível visualizar graficamente a condição da perfusão na zona do tecido em causa, o miocárdio, havendo uma conexão com o nível de oclusão das artérias coronárias.

Palavras chave: *Deep Learning*, Eixo Longo, Rede Neuronal Convolucional, Ventrículo Esquerdo.

List of Contents

List of Figures

List of Tables

List of Abbreviations

2 CH-LA	2-Chamber Long Axis
3 CH-LA	3-Chamber Long Axis
4 CH-LA	4-Chamber Long Axis
AI	Artificial Intelligence
AIF	Arterial Input Function
CAD	Coronary Artery Disease
CMR	Cardiovascular Magnetic Resonance
CNN	Convolutional Neural Network
CO	Cardiac Output
CPU	Central Processing Unit
CT	Computed Tomography
DC	Débito Cardíaco
DL	Deep Learning
EC	Eixo Curto
EDV	End Diastolic Volume
EES	Extravascular Extracellular Space
EF	Ejection Fraction
EL	Eixo Longo
ESV	End Systolic Volume
fCNN	Fully Convolutional Neural Network
FE	Fração de Ejeção
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HR	Heart Rate
IA	Inteligência Artificial
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IRF	Impulse Response Function
LA	Long Axis
LV	Left Ventricle
MBF	Myocardial Blood Flow
ML	Machine Learning
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
MT	Modelo de Tofts
NN	Neural Network
ReLU	Rectified Linear Unit
RMC	Ressonância Magnética Cardíaca
RMSE	Root Mean Squared Error
RNC	Rede Neuronal Convolucional
RNN	Recurrent Neural Network
ROI	Region of Interest
RV	Right Ventricle
SA	Short Axis
SD	Standard Deviation

SGD	Stochastic Gradient Descent
SV	Stroke Volume
TAC	Tomografia Axial Computorizada
TM	Tofts Model
VE	Ventrículo Esquerdo
VS	Volume Sistólico
WHO	World Health Organization

1 Introduction

When it concerns the performance of a certain cognitive task, the human brain and knowledge are the best options to go with. However, sometimes the task can be very demanding and time consuming and it would be beneficial to use a different tool allowing the human brain to do different things at the same time. AI was developed as an attempt to describe and simulate the human intelligence and learning process using computers.

After some development years, in the 80s and 90s researchers started to focus their attention on how the human brain works to learn how to reproduce its behavior as accurately as possible. That is when the first artificial neurons showed up creating the first artificial neural networks. These networks did not have as many neurons as the human brain but definitely had the power to solve problems such as classification and regression tasks, since the created networks could take conclusions from data, i.e. learn, this way creating the ML area. This new AI field started to produce breakthrough results, as it was demonstrated by the computer, Deep Blue, that beat the chess world champion, at the time, in a chess game.

In the mid 00s, a new field inside ML came up, the DL, where networks with more neuronal layers were created to solve more complicated tasks more independently than the ML algorithms, in a way similar to how humans think. With DL being a very powerful and desirable area to work on, its abilities started to be applied to different knowledge fields such as medical image processing.

This project was designed to explore the potentialities of the DL area in the medical image processing field. CNNs are made of artificial neurons layers connected between them that model the way the synapses occur in the human brain during the thinking process. These networks were used to analyze CMR images in a user independent way in order to segment and register them, using these results to quantify functional parameters related to the myocardial condition.

In Chapter 2 the background information regarding AI, ML, DL and cardiac functionality needed to fully understand this project's broadness can be found, following Chapter 3 with all the project's methodology used to achieve the results which can be found in Chapter 4. Chapters 5 and 6 contain the results' analysis and the conclusions taken from them together with some future work considerations, respectively.

2 Background

The present chapter focuses in introducing the essential concepts regarding cardiac medical images analysis and segmentation, cardiac performance parameters, ML and DL, and kinetic models, which will be essential to the understanding of these project's results.

2.1 Cardiac Performance and CMR Imaging

The heart is a muscular organ whose function is to receive the deoxygenated blood from the whole body, send it to the lungs in order to oxygenate and, once it is inside the heart again, send it to the whole body in order for the cells to exchange their carbon dioxide for the oxygen in the blood. The heart performs this action throughout the life span of an individual, cyclically, and, as in all mechanical systems, its performance must be evaluated when problems appear or, ideally, before these show up.

Usually these problems are related with cardiac conditions such as Coronary Artery Disease (CAD), which turns out to be one of the several cardiovascular diseases and the major cause of ischemic heart disease corresponding to the first cause of death worldwide, accordingly to the World Health Organization (WHO) [1]. CAD is characterized by the accumulation of atheroma plaques inside the coronary arteries, which reduce the blood flow to the heart muscle, the myocardium. This drop in the amount of blood that reaches the heart muscle creates a scenario of ischemia characterized by the lack of oxygen which can lead to myocardial infarction and, consequently, to death.

Diagnosis through CMR imaging is possible and its utilization is growing worldwide due to its non-invasive profile and no need for ionizing radiation nor iodine contrast [2]. To make this technique reliable to use it is necessary that it becomes easily implemented and relies on accurate and fast acquisition methods so the information derived from these images can be precisely interpreted. CMR is becoming increasingly popular when comparing to other imaging techniques because it can accurately measure: end-diastolic (when the heart at its most dilated phase) and end-systolic (when the heart at its most contracted phase) ventricular volumes, mass and also myocardial perfusion, i.e. oxygenated blood delivery to the myocardium, which is a very good indicator of CAD and cardiac performance [3].

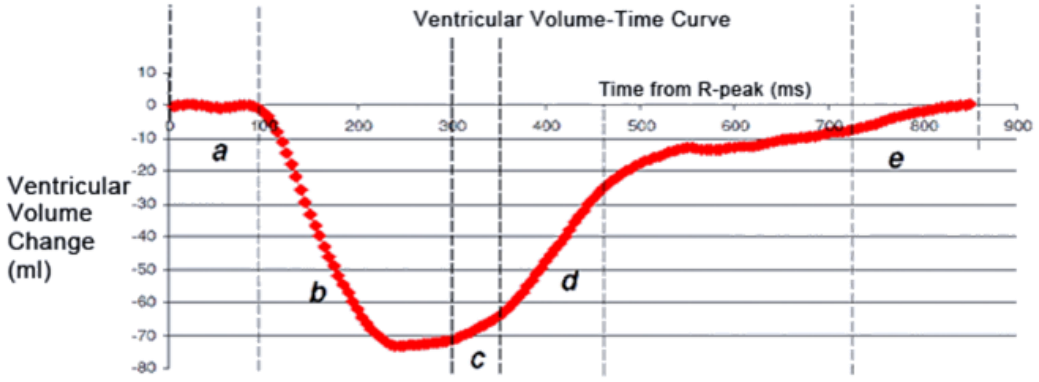
These different measures that can be obtained from CMR are strongly influenced by the used CMR modality since each one offers different information, with perfusion CMR being used to quantitatively assess the myocardial tissue perfusion level and functional cine cardiac Magnetic Resonance Imaging (MRI) being widely used due to the ability to visualize the contractile function of the myocardium. Consequently, cine cardiac MRI allows segmenting the myocardium and the LV and, also, facilitates the analysis of the myocardial contractile function by quantifying some parameters such as the SV, the EF and the CO [4] – [5].

2.2 Cardiac Functional Parameters

As previously mentioned the heart works in a cyclic way. The cardiac cycle corresponds to the time interval since the heart contracts, relaxes and contracts once again and can be split in two different

phases: ventricular systole and ventricular diastole. The systole is the contraction of the myocardium and correspondent blood ejection to the whole body, and the diastole the myocardial relaxation and correspondent ventricular filling.

Focusing on the LV, the ventricular systole starts with the mitral valve closure and left-ventricular pressure rise, but maintaining the ventricular volume, a phase known as isovolumetric contraction (Figure 2.1). When the pressure inside the LV is higher than in the aorta, the aortic valve opens and a fast blood ejection happens followed by a decrease in the ejection rhythm until the pressure in the aorta gets higher than the one in the LV, causing the aortic valve to close, the systole to come to an end and the diastole to start. When the aortic valve closes, the ventricular pressure decreases and there is a small increase in the LV volume, i.e. isovolumetric relaxation. The ventricular pressure continues to drop until the point when the mitral valve opens due to the pressure difference between the LV and the left atrium and the blood can flow inside the LV [6]. The ventricular filling is fast in the beginning, filling more than half of the ventricle, and then the ventricular pressure starts to rise causing the filling rate to decrease until the left atrium contracts and makes the rest of the blood flow from this cavity to the LV. As soon as the ventricular pressure overcomes the atrial pressure, the mitral valve closes and a new cycle can start with the systole phase. Figure 2.1 [7] shows the relation between the ventricular volume and time. It is important to mention that the same cycle happens for the right atrium and ventricle, at the same time. However, the ventricular volume – time curve is plotted regarding the LV as the blood goes from this cavity to the whole body, being possible to quantify some performance parameters.



a	0	–	100	ms	Isovolumetric contraction phase
b	100	–	300	ms	Ejection phase
c	300	–	350	ms	Isovolumetric relaxation phase
d	350	–	460	ms	Rapid filling phase
e	720	–	820	ms	Atrial kick

Figure 2.1 Ventricular volume – Time curve. The red line represents the LV relative volume changes over time. From the curve one can distinguish several phases from the cardiac cycle: a) represents the very beginning of the systole, i.e. the isovolumetric contraction phase, when there is no change in the ventricular volume in spite of the pressure increase; b) corresponds to the blood ejection phase which is fast in the beginning and then slows down until c) which represents the isovolumetric relaxation phase, i.e. a small increase in ventricular volume; d) translates the fast ventricular filling which gets slower as time goes on and e) shows the atrial kick which is the contraction of the left atrium in order to make the rest of the blood flow from the atria to the ventricle. Adapted from [7].

There are some physical parameters useful to evaluate the cardiac activity such as SV, EF and CO, as mentioned, and these parameters directly or indirectly depend on two quantities which can be

measured from the ventricular volume – time curve and, consequently, from CMR images: the End Diastolic Volume (EDV) and the End Systolic Volume (ESV).

The EDV is the volume of the ventricle at the end of diastole when this cavity if fully filled with blood just before systole begins and the ESV is the volume of the ventricle at the end of systole when the ventricle is minimally filled right after the blood ejection. Both these parameters can be calculated from the ventricular volume – time curve in Figure 2.2. This figure shows the relation between the ventricular volume with time. It represents the volume instead of the volume change (as in Figure 2.1), therefore EDV is the maximum value of the ventricular volume, which happens in the end of diastole, and ESV is the minimum value of the ventricular volume, which happens in the end of the systolic phase.

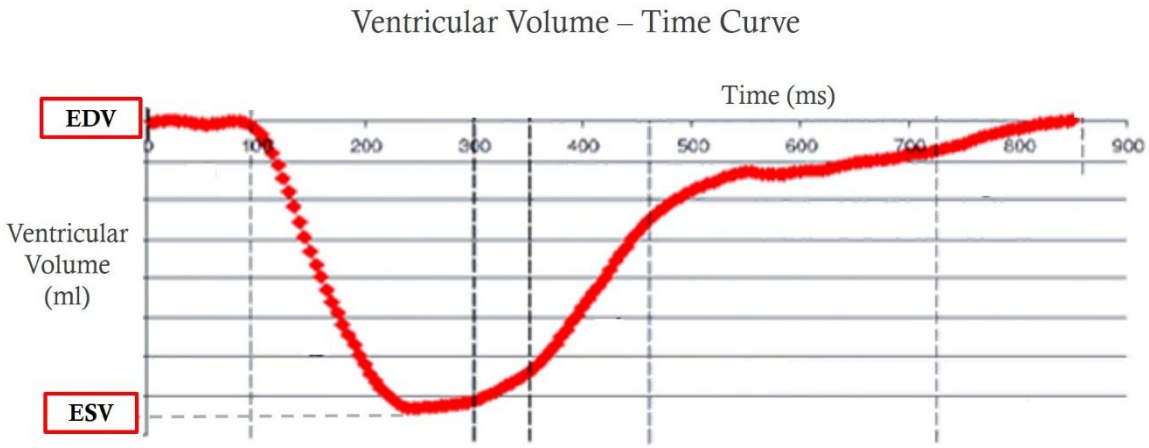


Figure 2.2 Ventricular volume dependency over time. Please note that the vertical axis represents the absolute ventricular volume. EDV represents the volume when the ventricle is fully filled, corresponding, in the graph, to the maximum volume possible. ESV represents the volume when the ventricle is minimally filled, corresponding, in the graph, to the minimum volume possible. Adapted from [7].

From the EDV and ESV one can directly get the SV. It represents the total amount of blood that is ejected during one myocardial contraction and it can be quantified from:

$$SV = EDV - ESV \tag{2.1}$$

The EF translates the fraction of EDV that is ejected from the ventricle during one myocardial contraction. This parameter is an indicator of the heart's contractility and can be derived from the SV and the EDV accordingly to:

$$EF = \frac{SV}{EDV} \times 100\% \tag{2.2}$$

Finally, the CO is the total amount of blood ejected from the heart in one minute and this measure is a good approximation of the cardiac function. It comes as in equation 2.3 where HR represents the Heart Rate:

$$CO = SV \times HR \tag{2.3}$$

Accordingly to [6], the normal range for each of these parameters is represented in Table 2.1, considering a healthy subject.

Table 2.1 Normal range values for cardiac performance parameters and heart rate, considering a healthy subject.

Parameter	Normal Range
SV	50 to 100 mL
EF	55% to 75%
CO	1.9 to 3.5 L/min
HR	60 to 70 beats/min

As previously mentioned, CAD leads to ischemia which can lead to infarction if the oxygen supply to the affected area is not restored. The areas that suffer ischemia are still considered as living tissue but as soon as the infarction starts, the tissue will start to die and, in the cardiac scenario, the myocardium will lose its ability to contract creating non-functional areas in the heart muscle. These deceased regions will affect the cardiac performance and, therefore, the EDV, ESV, SV, EF and CO final values, allowing to differentiate healthy from non-healthy patients by plotting the ventricular volume – time curve and quantifying the parameters.

2.3 Heart Segmentation

When evaluating cardiac performance, it is useful to accurately localize and segment not only the LV but also the myocardium, which is responsible for the heart contraction. Both the epicardium (the thin serous membrane which protects and lubricates the outside of the heart), and the endocardium (the squamous layer that keeps the blood flowing avoiding it to stick to the inside of the heart), need to be segmented in order to retrieve the parameters related to the myocardium, like its thickness and perfusion levels, among others [4]-[5]. When obtaining certain parameters such as SV, EF or CO, there is not the direct need to segment the myocardium since the LV segmentation is enough [8].

There are several views of CMR images (Figure 2.3) from which one can localize and, consequently, segment the LV, such as SA, 2 CH-LA, 3 CH-LA and 4 CH-LA. The existing segmentation approaches are mainly based on SA images, where the LV and the myocardium can be segmented with a good level of accuracy, since the SA plane is the most commonly used one when working with cardiac images [9] and is also considered as a gold standard. Also in this plane both the myocardium and the LV are easier to detect due to geometric properties. However the SA images showing the heart apex and the valve plane present some difficulties when trying to reconstruct the LV

volume from a SA cine sequence. To avoid this influence in the LV volume value, this can also be quantified from LA images, where is also easy to segment the LV.

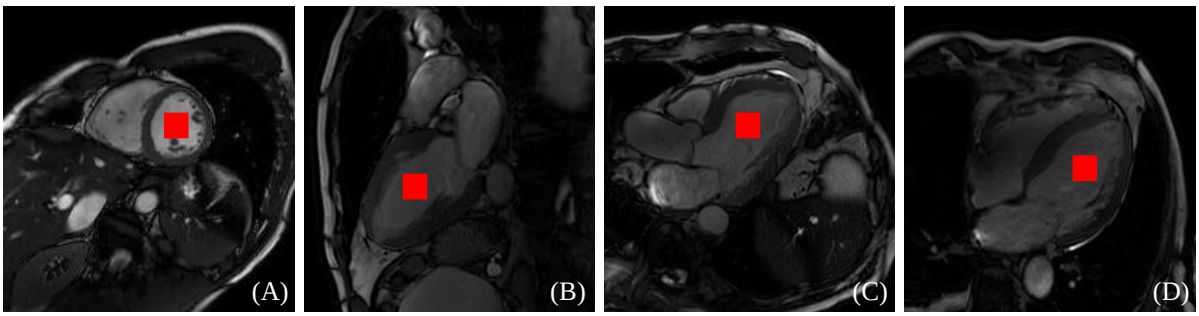


Figure 2.3 Different views of CMR images, where the LV is pointed out with the red square: (A) SA view, (B) 2 CH-LA view, (C) 3 CH-LA view, (D) 4 CH-LA view. Images obtained from the data set used in this project.

The most accurate and precise way to segment both the LV and the myocardium is still manual, being the gold standard in clinical practices and in image processing and being widely used to evaluate the accuracy of fully automatic segmentation methods. However, manual segmentation is very time consuming, user dependent [10] and error sensitive. Consequently, there is a need for automatic LV segmentation approaches, which are significantly challenging mainly because the current existing approaches still require some manual corrections or some initial data processing in order to establish a Region of Interest (ROI). Additionally, the presence of papillary muscles, which connect the apex of the LV to the mitral valve controlling its opening and closing, also influences the final segmentation, due to the similarity between the grey levels of these muscles and the myocardium.

The majority of the existing segmentation methods are based on (1) LV localization, (2) time-dependent, and (3) shape-dependent techniques. The approach (1) assumes that the heart is in the center of the image but does not take into account its variability of size and spatial orientation from one patient to another [11]. Approach (2) assumes that the heart is the only organ moving in the image but it can fail due to poor sensitivity as, for example, the lungs also move and produce image artifacts that must be considered. Approach (3) considers the LV a circular shape but may fail when the ventricles have abnormal shapes or when dealing with LA images, where the LV has more of an elliptical shape than a circular one. Despite all these slight shortcomings, many of the existing segmentation methods are based on SA images and can be quite precise. However the usage of LA images can achieve great results similar to when using SA images [12], despite being a less common approach.

Recently, a set of methods collectively named as DL have advanced the state of the art in many computer vision and image analysis tasks, including medical image analysis. These new methods rely on CNNs, to achieve a fully automatic segmentation method that accurately localizes and segments the LV, and, in the end, does not require any manual corrections.

2.4 Machine and Deep Learning

AI is a wide region of knowledge inside which one can find the ML area. A ML approach, in practice, analyzes a set of data and all the features that are linked to this data set (*a priori* information),

learns from it and, in the end, makes a prediction about an unseen observation (Figure 2.4). For example, if the data set is made of annotated functional CMR images, the ML approach learns features previously obtained from those images (the training set) to make correct predictions about unseen CMR images (the test set). However, if the new images in the test set were acquired using a different methodology to the one used to obtain the training data set or if they present a significant difference from the ones in the training set, for example in contrast or intensity, the performance of the ML model is likely to be compromised and yield low accuracy levels, since the used features during the training process are no longer reliable to classify the new data set and it is necessary to extract other features. Still regarding the ML model, the training process includes, in most of the times, manual feature extraction, i.e. the user helps the network to look for the characteristics it considers more important for the final output.

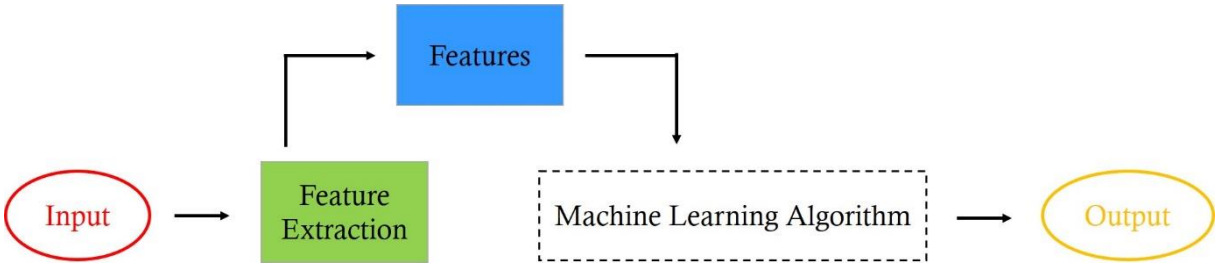


Figure 2.4 ML working process. The features are extracted *a priori* from the inputs and then are used by the ML algorithm to correctly predict a certain input, creating an output. Usually the feature extraction is manual, requiring a user to do it.

The DL concept, which creates a domain included inside the ML one, appeared to define a set of methods that, despite sharing the same working principle as machine learning ones, i.e. learn from data and make a prediction, are inspired by the human brain behavior and are capable of learning features from raw data by themselves, i.e. without user influence (Figure 2.5). Besides this last-mentioned advantage, DL methods are more powerful, therefore requiring more computational power and the utilization of a GPU (Graphics Processing Unit) instead of the CPU (Central Processing Unit), can handle big data, are more prepared to deal with images and can show a good performance even when there is variability among the data, for example CMR images obtained with different MRI scanners, but at the cost of a greater training time.

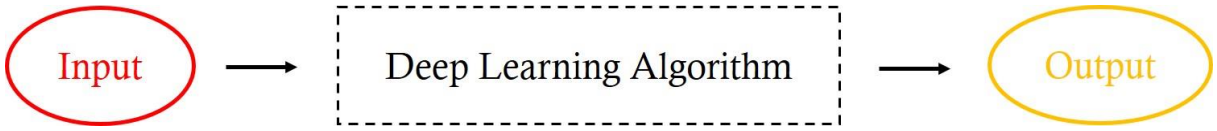


Figure 2.5 DL working process. With this new set of models there is no need to extract features prior to the usage of the DL algorithm, since it has the ability to do it by itself.

There are three more common ways to make machine and deep learning algorithms learn something: (1) supervised learning, (2) unsupervised learning and (3) semi-supervised learning. Approach (1) happens when the algorithm tries to establish a relationship between the given input and the output with this relationship being evaluated as the algorithm learns since it uses labeled data sets, i.e. the algorithm knows the correct output for each input. Approach (2) is followed when the algorithm

learns or looks for patterns in unlabeled data which seem more relevant to assess the output even though there is not a correct or wrong output since there are no labels. Approach (3) happens when the algorithm learns from a data set that contains both labeled and unlabeled entries [13], and in this case one can use both supervised algorithms to try to predict labels for the unlabeled data and unsupervised algorithms to try to find a pattern between the variables. Supervised machine learning (1) usually obtains the best results but big sets of labeled data are very difficult and expensive to get. Therefore, unsupervised methods (2) are also used with results not as good as supervised learning ones but which have been improving.

In a general way, ML algorithms (where DL ones are included) are made up of three important parts [14]: (1) the system that makes the predictions, i.e., the model that outputs the final result. It can go from a simple binary classifier distinguishing between 2 different classes (healthy or non-healthy) to a more complex system that segments a CMR image, for example. Parts (2) and (3) are the learning algorithm and its parameters, θ, respectively. The learning algorithm uses the features previously collected in the ML case or learns them from the training data set, X, in the DL case, and adjusts the parameters, θ, so the output, i.e. the prediction, $\hat{Y}$, provided by part (1), matches the input, i.e. the ground truth, Y, as accurately as possible. These last steps of adjusting parameters and improving the learning algorithm are usually referred as "learning a model" or "training" it.

2.5 Learning a Model

Learning a model (Figure 2.6), is an iterative process and it is carried out following some essential steps, starting with the definition of a score function (done by the network), moving on to the choice of a loss function and ending with an optimization problem [15]. The score function establishes a score between the images and all classes being considered, accordingly to the function parameters. Considering a simple binary classifier, this model relies on very simple and not complex functions, such as a linear regression (equation 2.4). In a binary classifier, considering a set X with N images, the score function associates one value to each pixel from each image i:

$$\mathrm{sf}\,(X_i, W, b) = WX_i + b \qquad 2.4$$

This simple linear score function, sf, only has 2 parameters: the weights, W, which interact with the images, X_i, and the biases, b, which do not interact with the images but influence the final value for each one, as the biases reflect the learner's ability to keep learning something wrong. W and b can be referred to just as weights or, more simply, parameters. When considering a more complex model to make a prediction, like in DL models, the used score functions still attribute a value to each image/classes pair, but these functions are more complex and contain more parameters because there are many more images being used and, usually, one is not dealing with binary classes.

The loss function, L, measures how good the model being learned is when it actually performs its predictive task. Since it is not possible to change X, one can only manipulate the parameters, θ, and change them so that the predicted label for each image, $\hat{Y}_i$, matches the ground truth one, Y_i. Therefore, the loss function, now $L(\theta)$, measures the quality of the parameters by quantifying the difference

between the real training labels and the ones obtained during the learning process (equation 2.5), with E being the number of iterations during the learning process.

$$L(\theta) = \sum_{i=1}^{E} \left[Y_i - \hat{Y}_i(X_i, \theta) \right]^2 \qquad 2.5$$

If the prediction made by the model does not match the reality ($Y_i \neq \hat{Y}_i$), then the loss will be high and, on the other hand, if the obtained prediction is close to reality ($Y_i = \hat{Y}_i$), the loss will be low. Following the previous example of the binary classifier, the loss function is such that it considers that some label prediction is correct if the attributed score to the correct class is higher than the score of the wrong class by a certain threshold.

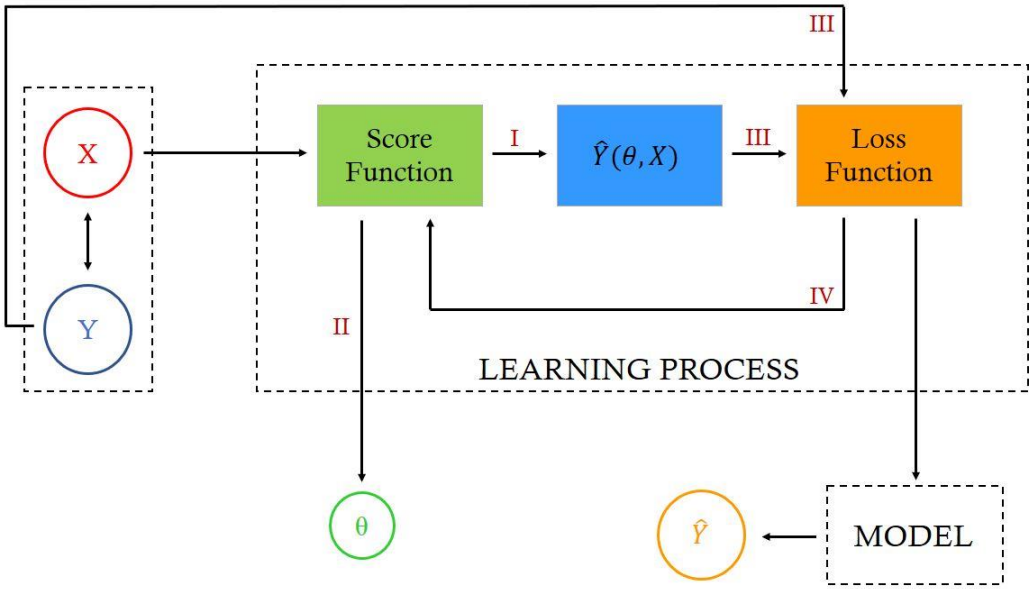


Figure 2.6 Iterative process of learning a model. X and Y represent the training images set and its labels, respectively. The Model, Learning Process and Parameters are the three major parts of a ML system: the score function receives as input the images X, and its outputs are – the parameters θ (II) and, accordingly with the produced score for each image, a prediction $\hat{Y}$ (I) of the corresponding label, the loss function receives (III) the previously produced prediction, $\hat{Y}$, and the real label, Y, corresponding to the image being analyzed, and compares them producing a loss value which is used as input in the score function (IV) allowing to update θ and produce new $\hat{Y}$. This iterative process, which reflects the optimization problem, continues until the loss is minimum and the predictive model outputs the final prediction for the image label.

Since that the loss provided by $L(\theta)$ should be minimal, i.e. the prediction should match the ground truth, the optimization problem comes up. Optimizing a model is looking for the best parameters set θ, which is the set that minimizes the loss. This optimization in DL models relies on powerful methods such as Stochastic Gradient Descent (SGD).

All this learning process (Figure 2.6) is performed using the training set allowing, in the end, to save the parameters and then use the learned model on the validation and test sets. This is the reason

why this learning process is also called training: one trains the network based on a training set, saves the parameters and the overall model so it can be used again without the need to train it one more time.

2.6 Optimization and Stochastic Gradient Descent (SGD)

Like it was mentioned, optimization is a fine way to find the proper parameters set, θ, and this is the set that minimizes the loss function. The first thought might be to try and use different values for the parameters and keep track of the ones which provide the minimum loss but it is clear that this approach is not the best one since it is time consuming, especially if there are many parameters to find, very rudimentary and random. The best way to get the best parameters and the lowest loss as possible is to use iterations, i.e. start with random values and change them in an iterative way [16].

The concept behind the iterative idea is based on the usage of gradients, i.e. derivatives. In the described optimization scenario, the loss function gradients are used, since these indicate the direction of this function maximal variation, introducing the Gradient Descent algorithm which tells how to change the input in order to improve the output.

The SGD algorithm works as follows: the loss function gradients with respect to each parameter are calculated and then each parameter update is set at every iteration by evaluating the gradients (gradient descent). The gradients point in the direction of the function maximal variation but since the goal is to minimize the loss, the algorithm must follow the opposite direction, i.e. the opposite sign of the gradient.

When following the chosen direction, the algorithm moves along it in several steps whose size is determined by the learning rate. The learning rate tells how far should the algorithm advance at each iteration in order to achieve the lowest loss. It can be set in different ways but the most common one is to set the learning rate as a small constant since it is better to take a little longer to find the optimal solution than to set the learning rate too high, giving a larger step that leads to faster progress at the cost of maybe missing the optimal point. This parameter can, later, be updated to its best value.

The optimization algorithm computes each parameter update based on the loss obtained using only a small sample from the training set, i.e. a minibatch, at each iteration, instead of doing it for all examples in the data set individually, since the gradient obtained from a minibatch is a good approximation of the gradient obtained from all examples in the training set [16]. This type of algorithm is called stochastic since it only uses minibatches [17]. Usually the minibatch size is a small number between 1 and 100 and in medical images datasets it does not get greater than 10. Another SGD property is that often the training sets are very large and contain thousands of examples which forces the SGD algorithm to perform more iterations until it reaches convergence, however SGD will always converge before it analyzes all examples in the training set and perform parameter updates more frequently (minibatch advantage).

2.7 Training and Validation Errors and Regularization

When creating a DL model there is no certainty that it will work properly when analyzing images never seen before, i.e. test sets, therefore generalization can be defined as the system ability to work well on test sets, and a new error can be quantified, the generalization error. Optimization focuses in

minimizing the training error, i.e. error measured using a training set (loss is a good measure of this error), and, on the other side, the way to minimize the generalization error relies on regularization methods [17]. The test set is a precious resource that should only be used once, in the end, when the DL algorithm performance cannot be further improved. Therefore, the test error can only be quantified in the end. To avoid reaching the test phase without having the best model, i.e. the one that has the best performance, as possible, the regularization error is calculated from the validation set. It is important to keep in mind that the regularization error will always be greater than the training one and, in the best-case scenario, they will be equal, and the relationship between these two errors must reflect 2 properties: (1) the training error must be as low as possible and (2) the gap between the training and regularization errors should be as small as possible. When these conditions are not achieved by the DL algorithm performance, 2 scenarios can occur: underfitting and overfitting. Underfitting happens when the training error is not the lowest, i.e. property (1) is not fulfilled, and overfitting occurs when the gap between both errors is too large, i.e. property (2) is not fulfilled (Figure 2.7).

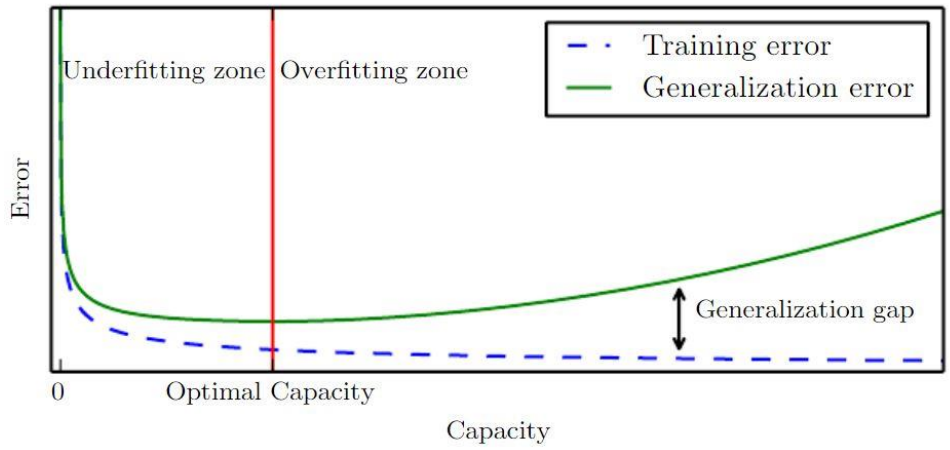


Figure 2.7 Graphic representation of training error (dashed blue line) and generalization error (green line), in terms of model capacity, underfitting and overfitting. On the left side of the red line the underfitting domain is displayed as both training and regularization errors are high and on the right side one can see the overfitting domain where the gap between both errors is increasing. Adapted from [17].

In case of overfitting the learning model capacity should be stopped at its best value (Figure 2.7 – red line). The model capacity is translated by the degree of the polynomial that best fits the training data so when a model has a low capacity, like a binary classifier whose capacity is 1, it cannot fit a wide variety of data and when the capacity is too high the model fits too perfectly to the training set data not being able to perform very good on test sets, since these are different from the training ones, consequently leading to overfitting. A good way to change model capacity, and avoid overfitting, is by setting a hypothesis space made of a limited number of functions that the model can use during the learning process which also limits the number of parameters θ, which is another factor contributing to overfitting. As the amount of data in the training set increases, the model being learned becomes more and more complex using more complex score functions and, therefore, computing more parameters needed to make the final prediction.

As previously mentioned, regularization methods influence the parameters of the model, θ, keeping these as low as possible and even making some of them equal to zero. Regularization is important to obtain good parameters, which lead to a better predictive model and, consequently, to a better prediction, and also to avoid overfitting [15], therefore regularization is another way to influence model capacity.

The regularization usually is included in the loss function, influencing its outcome, in many different ways as there are many different regularization methods.

Dropout is one form of regularization and it is described as a method to reduce interdependent learning between the neurons of the neural network, i.e. during the training phase some randomly chosen neurons are not considered [18] since the neurons belonging to a certain layer can depend on each other therefore sharing the same information and overloading the final set of parameters (Figure 2.8). Dropout works in a probabilistic manner where there is some probability, p, of a neuron being ignored at each layer.

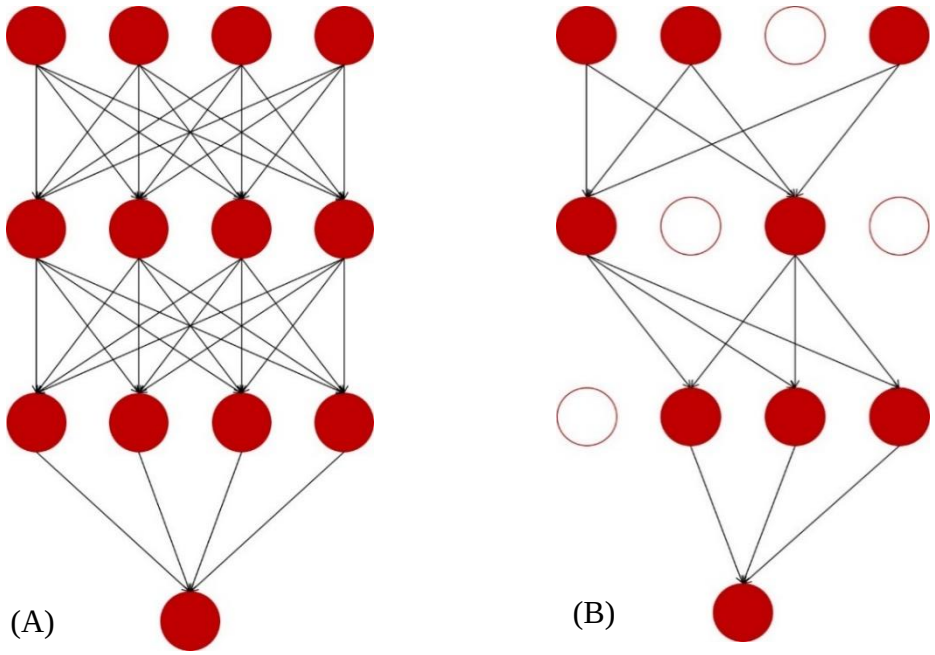


Figure 2.8 Dropout scheme. (A) represents a simple neural network where all neurons are working and sharing weights between them, all of them contributing for the final output of the model. (B) represents the same neural network but with the regularization method known as dropout. The white neurons are ignored, what creates less connections between the neurons reducing the interdependent parameter learning.

Another common regularization method is the L2-Regularization, also known as Weight Decay, which influences the loss function as shown in equation 2.6.

$$L(\theta) = \sum_{i=1}^{N} \left[Y_i - \hat{Y}_i(X_i, \theta)\right]^2 + \frac{1}{2}\lambda\theta^2 \qquad 2.6$$

The new added term is called the regularization term or the regularizer, and it strongly penalizes the model complexity. The constant λ reflects the preference for larger or smaller θ: a larger λ makes the parameters go smaller so the loss can remain the same (ideally as small as possible), but not excessively large so the regularizer does not become more important than finding the parameters that provide the lowest lossand a λ which approaches zero will lead the model to overfit. λ is called the regularization strength [16] and its initial value is unknown, so this should be picked in the beginning,

not at random but from literature recommended values, avoiding very large or very small values, and then perform cross-validation on it [19] in order to find its best value.

2.8 Hyperparameters and Algorithm Performance

The parameter λ is not computed by the DL system, it is, instead, selected by the user and then updated. All variables that are used to control the deep learning system performance, such as λ, the model capacity or the dropout percentage, and are not learned by the algorithm are known as hyperparameters. These hyperparameters are updated using the validation set and not the test set, otherwise the hyperparameters values would be part of a model that would work perfectly when considering that specific test set, leading to overfitting. The validation set is, consequently, used to set these hyperparameters.

Cross-validation is performed when the validation set is small or even nonexistent, making the hyperparameters initialization and update complicated [20]. During cross-validation instead of creating a validation set from the training set, one splits the training set in a certain number *k* of equal folds and uses one of them for validation and the rest for training, iterating over which fold is used for validation. For each value of the hyperparameter one wants to quantify, it is trained in *k-1* folds and validated in *one*-fold changing the validation fold at each iteration, creating *k* cross-validation accuracies and the chosen value is the one that provides the highest accuracy. However cross-validation is computationally expensive and one validation fold is preferred.

The learning rate and the minibatch size are, as well, hyperparameters that need to be set before the training starts. However, it is not common to cross-validate the later since its value is constrained by memory issues and computation time.

To make sure the used DL algorithm is as good as possible, the validation set is also used to obtain the validation error which corresponds to the training error but regarding the validation set. Given a training data set, one initially obtains certain values for the parameters θ (Figure 2.6 – path II) which, then, allow to obtain a prediction $\hat{Y}$ (Figure 2.6). With these predictions and parameters the loss can be quantified (the loss acts as the training error as it measures how different the prediction is from the ground truth). During the training, the model is also used to make predictions on the validation data set which will allow to quantify the validation error. Having both training and validation errors, they can be compared following the conditions presented before: (1) the training error must be as low as possible and (2) the gap between the training and validation errors should be as small as possible. The algorithm training is performed until the commitment between both conditions cannot be further improved. The final trained and validated model can then be used and its performance evaluated on a test set [15]. As previously mentioned, the test set should only be used at this stage when one has the best predictive model (with proper losses and regularization).

2.9 Convolutional Neural Networks

The task of recognizing something in an image presents lots of challenges, which influence the DL model final result, such as: scale, illumination and viewpoint variations, deformations, occlusions and

also intra-class variations. The algorithm used to perform the predictive task must be invariant to all these aspects [16] so it can be generalized to a wide variety of data sets.

DL algorithms use CNNs which work like the neurons in our brain. In CNNs there are layers composed by artificial neurons connected to adjacent layer neurons but not connected between them in the same layer and are explicitly used when the inputs are images, allowing to extract more features since these networks are made of more kinds of layers and neurons and use more complex mathematical operations, such as convolution. In the simple Neural Networks (NNs) every neuron from one layer is connected to all neurons of the next layer (Figure 2.9 – A) and these networks do not share weights between layers, creating a great number of numerical parameters to describe the predictive model. Regarding CNNs, these share the weights so the model does not have to learn different features every time to detect the same object in one image, this way reducing the number of parameters that need to be learned, and the neurons from one layer are only connected to some of the next layer (Figure 2.9 – B). This architecture is based in biological processes such as the ones that occur in the visual cortex, where individual neurons respond to stimuli in a limited receptive field. These receptive fields overlap, so the full field, like an image, can be fully processed.

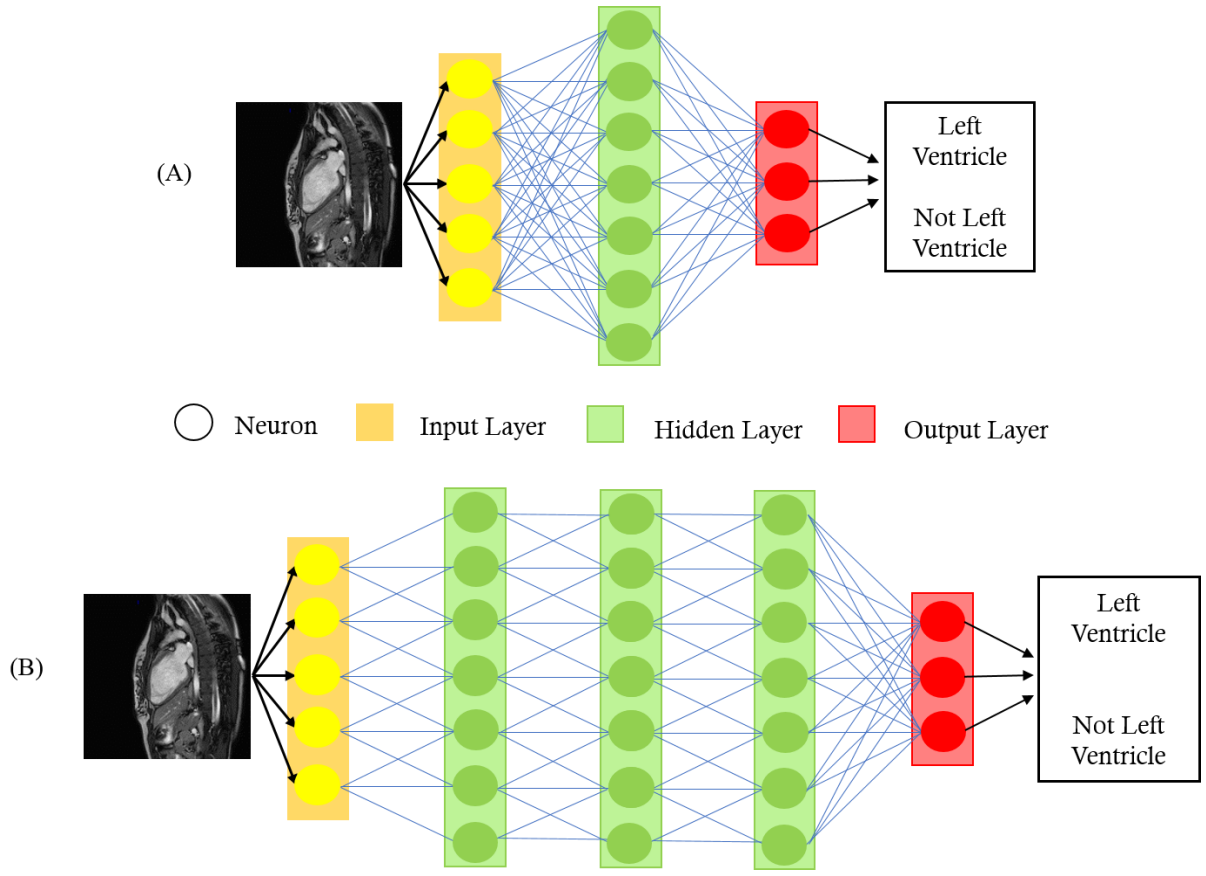


Figure 2.9 (A) Schematic representation of a simple NN, usually used in ML (requires more *a priori* information). (B) Schematic representation of a DL NN (requires less *a priori* information due to the greater number and variety of layers).

CNNs have different kinds of layers such as (1) input layer that receives the input image, (2) convolutional layers that compute the convolution (dot product) between neurons' weights and the input region they are connected to, (3) Rectified Linear Units (ReLU) which are nonlinear activation functions

that allow to obtain an output given a certain input, and (4) pooling layers, typically max-pooling, which aggregate pixels from the neighborhoods of the image area being analyzed whose value is maximal (max-pooling case), reducing the image size and, also, the number of parameters in the model. There are also (5) fully connected layers that compute each class score and in these layers each neuron is connected to all neurons from the previous layer, and (6) classification layer that uses a function to predict the class accordingly to the scores from the fully connected layers, being softmax the most commonly used. Layers 3 and 4 implement their own functions and layers 2 and 5 depend on their neurons' parameters and on the activation function applied to the input.

As mentioned, the activation functions are nonlinear, otherwise the network would behave like a linear regression model with limited learning ability. Since DL is more complex and uses more complicated datasets, nonlinear functions are used since they offer the network the power to learn any function that best maps the inputs to the outputs. Each neuron computes the dot product between its weights and the input, adds the biases and introduces the nonlinearities through the activation function (Figure 2.10). There are several options to use as activation function though recently ReLUs became popular and are widely used in DL models [21]. Equation 2.7 shows the ReLU function which only activates a certain neuron, *z*, if its output is positive.

$$f(z) = \max(0, z) \tag{2.7}$$

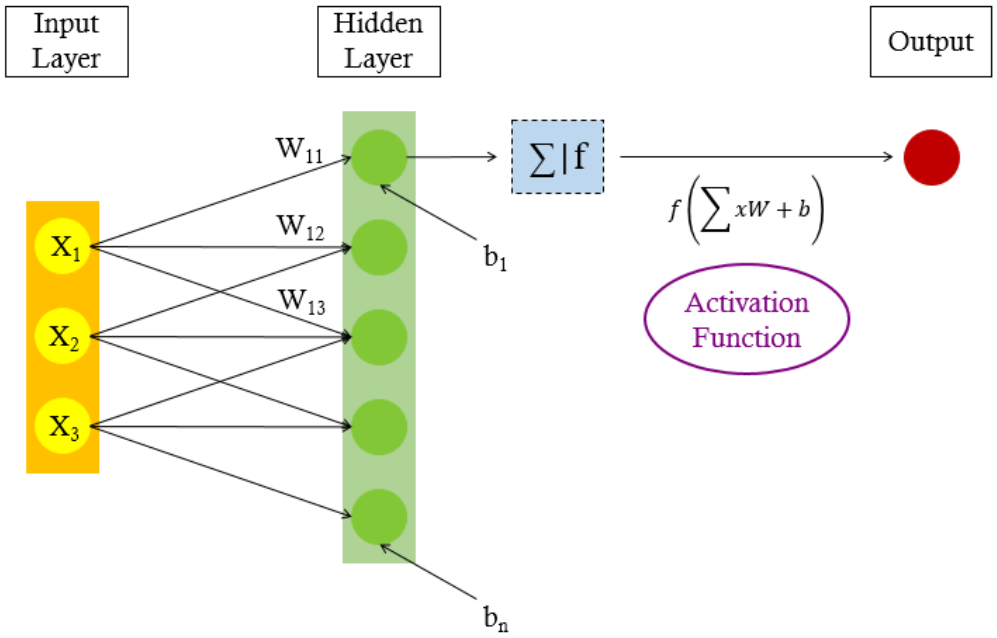


Figure 2.10 Activation function influence on a CNN. Using the input, *X*, and the layer weights, *W*, the convolution between these two is calculated and the biases, *b*, are added. The output of these operations, considering one neuron at the time, is submitted to the activation function, *f*, judgment which decides to use this neuron output as input to the next layer or not. The working principle is analogous for the rest of the hidden layer neurons.

The CNN behavior can be explained accordingly to Figure 2.11: the network receives an image as input, it enters the first convolutional layer, which has ReLU as an activation function, and the convolution is performed accordingly to the kernel used by the layer. The kernel is the filter that the

layer will use to extract the feature that it considers more relevant and this kernel can have different sizes, which can be specified by the user along with the number of filters used by each layer. Therefore, the convolution is the dot product between the kernel and an area of the input image whose size corresponds to the kernel size (blue square in Figure 2.11). When the image is larger than the kernel, the kernel receptive field does not cover the full image and several convolutions have to be performed moving the kernel receptive field along the image and the result of this operation is translated into a feature map. There are as many feature maps as kernels used. Once these maps are obtained, the pooling layer aggregates the pixels from the image neighborhood reducing their size and these new feature maps will be the input of the next convolutional layer where convolutions will be, again, performed accordingly to the new kernels size and number. After this feature extraction process, the fully connected layer analyzes all feature maps computing a score for each one, and the classification layer, represented by the softmax function, converts this score into a class.

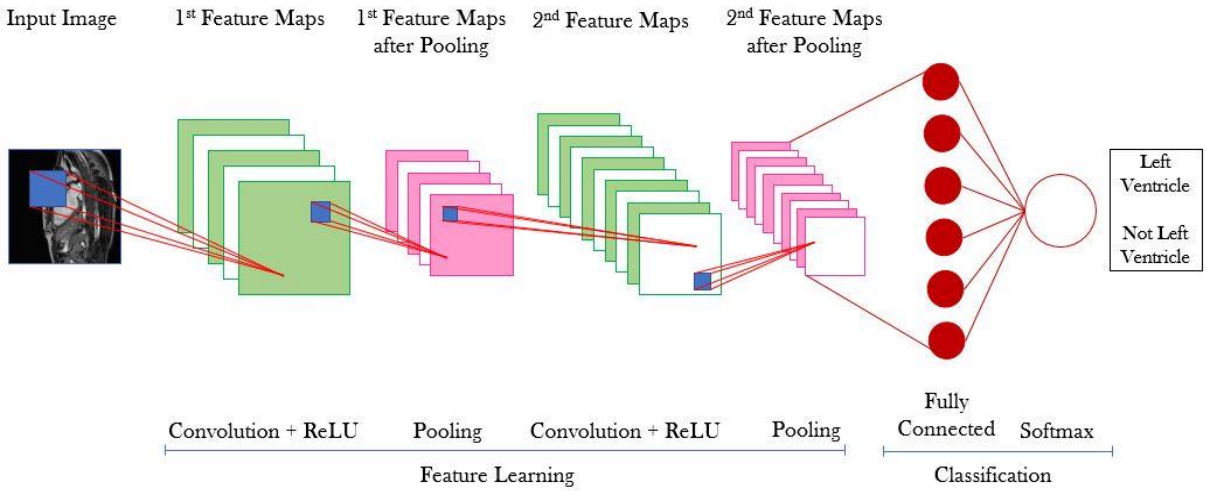


Figure 2.11 CNN with the input layer, 2 convolutional layers with ReLU, 2 pooling layers, 1 fully connected layer and a classification layer using softmax function to analyze the scores from the fully connected layer and provide the class output. The blue square represents the receptive field of the considered kernel and depends on its size. There are as much feature maps after each convolutional layer as kernels used.

The softmax function used in the classification process analyzes the probabilities for all classes and allows to classify them. The main advantages of this function are that all probabilities range from 0 to 1 (normalized class probabilities) and the sum of all classes probabilities is 1, making this the most used function by neural networks [16]. Softmax is widely used in multi-classification tasks since the other option, the sigmoid function, only works for binary classification problems because it only provides the probability for one class.

The chosen loss function depends on the classification task. When using the softmax function for classification, the used loss function is cross-entropy. This function is more complex that the one represented by equation 2.5, even though it still measures the distance between the real labels and the ones outputted by the model. The cross-entropy function is defined in equation 2.8, where p_i represents the real label and q_i the predicted one for the i^{th} image.

$$H(p,q) = -\sum_{i} p_i \log q_i \quad 2.8$$

This way, the softmax classifier minimizes the cross-entropy between two label distributions: the real one, p, and the estimated one, q.

During the network training each layer input is always changing, a process called covariance shift [22], since the minibatch and the previous layer parameters change. This large variability slows down the training and produces a large number of parameters, using too much memory space, therefore an operation is performed in order to reduce the internal covariance shift, the batch normalization. The batch normalization [23] normalizes both the input and the hidden layers' inputs to improve the training speed and makes each layer slightly more independent from the previous ones when learning the model, demonstrating some regularization power (similar to dropout), reducing the overfitting.

The number of layers of a CNN, excluding the input layer and the output, represent the model depth, i.e. the more layers it has, the deeper it is, what brings more complexity but also achieves better results. In the convolutional layers, using too many kernels can lead to overfitting, because the model is learning the same things over and over. That is another reason to perform validation. Conversely, the more specific the objects that one is trying to detect in the image are, the more kernels are needed, so there must be a good balance between overfitting and the level of details one wants to detect.

CNNs are the most used DL architectures in image analysis, including medical ones. DL approaches have some cons like the requirement of a big amount of training data, the consequent need of more computing power and also the complexity of the CNN architecture, but the fact that these approaches become more powerful than ML ones, perform better on difficult learning tasks, the feature extraction from the input images is more accurate and does not require *a priori* information, and the network architecture can be easily modified and adapted to new tasks, makes of the DL algorithms a growing method to automatically segment images.

2.10 Classification and Regression Scenarios

So far a classification scenario has been presented to introduce the most important concepts in machine and deep learning. The straightforward difference between classification and regression is that the first predicts a label and the second predicts a quantity.

As presented, in classification scenarios the outputs are discrete, i.e. classes. The number of classes can go from 2, i.e. binary classification, to as much as desired, i.e., multi-class classification. Usually these classification algorithms predict a continuous value for each class probability and, by setting a threshold, one can choose which class matters the most for the final solution. Differently, in regression models the outputs are continuous, i.e. they predict an integer or a float value.

However, in both scenarios, the inputs can be real numbers, i.e. continuous, or discrete variables and NNs can be used for classification and regression providing some minor changes in the network structure, especially in the number of hidden layers, which is lower in regression, in the convolution dimension, and in the considered loss function. Usually in regression the most commonly used ones are the Mean Squared Error (MSE), equation 2.9, or the Root Mean Squared Error (RMSE), equation 2.10.

Both of them are error measures and measure how spread around the optimal fitting line the residuals, i.e. the predictions, are, however RMSE penalizes large errors and does not describe the mean error as well as MSE.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{Y}_i - Y_i\right)^2 \quad 2.9$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left(\hat{Y}_i - Y_i\right)^2}{N}} \quad 2.10$$

2.11 Deep Learning Progresses and State of the Art Results

Nowadays ML and DL methods are widely spread and used in medical image analysis. This review covers the most relevant studies and results in ML areas and their applications in medical image analysis, starting with the, now outdated, computerized methods to analyze images, following the rising of ML, the best results obtained in this area and the efforts to overcome certain limitations, moving on to the use of CNNs and finishing with a more complete review of the application of these studies in cardiac image analysis.

Initially, computerized techniques were mainly used to process images and perform some mathematical modeling such as fitting circles and lines to relevant structures. In the 90s ML approaches started to become popular, specifically the supervised techniques, i.e. when the user guides the algorithm, such as active shape models used to segment images and feature extraction algorithms to help doctors when making diagnosis. Kass et al. [24] successfully used snakes, i.e. active contour model, to interactively establish image contours by minimizing an energy function. Later, Lobregt et al. [25] developed an active shape model described in terms of forces and its fields rather than energy functions like snakes model. This group work concluded that the final contour draw depends on the image features used to guide the forces. More recently, Hautvast et al. [26] used the same model developed in [25] adapted to propagate left ventricular contours over a cine CMR sequence which can be used to quantify cardiac parameters such as SV, CO and EF.

It is known that the extraction of relevant features from images affects the final output of the ML model. Therefore, the need for an automatic system capable of extracting the correct features from an image existed and DL algorithms started to come up to solve this need. These DL algorithms are networks made of more and different kinds of layers that receive the input, provide an output and on the meanwhile learn features considered more discriminant to correctly classify the input. This development of models that can extract features on their own was gradual. In the beginning feature extraction was manually engineered, a process that is time consuming and depends on the user experience and knowledge, and before using DL models some other techniques were developed like principal component analysis and images clustering. The latter techniques are examples of unsupervised learning, i.e. when the algorithm learns or looks for patterns in the data which seem more relevant. After these two unsupervised learning techniques and a period where efforts were done to develop the DL area but no significantly results were achieved, a breakthrough appeared, the AlexNet, based in the previous

work of LeCun et al. [27] during the 90s. [27] developed LeNet, a CNN with less convolutional layers than AlexNet, therefore less deep and less powerful, but powerful enough to achieve promising results classifying images with numbers.

Krizhevsky et al. [28] won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), whose goal is to evaluate algorithms used at large scale for image classification and object detection, by a great margin, using AlexNet, a large and deep CNN with 5 convolutional layers, max-pooling ones and 3 fully-connected layers at the end, to produce the correct output. After this good result in image classification and object recognition, DL became the chosen technique to extract relevant features from images and to use on medical image analysis scenarios, using supervised training.

More recently Goodfellow et al. [29] created a new framework capable of learn features from images, in an unsupervised way, and generate new images. It has a generative model that generates new samples from the input images and a discriminative part that distinguishes between real and generated samples. Chuquicusma et al. [30] used this Generative Adversarial Network (GAN) to create new lung nodules samples later classified by radiologists and the result was that these professionals could not distinguish between a real and a generated sample, showing that the use of GANs might be useful to produce big data sets and also to train medical professionals.

In the beginning of the current decade the need for deeper models showed up because AlexNet and LeNet were no longer enough. Instead of creating networks that use kernels with a large receptive field, new networks came up with smaller kernels whose utilization reduced the amount of memory needed. Simonyan et al. [31] were the first to explore this technique of deeper networks using several small kernels in each layer and created VGGNet, due to the good results. The winner of the ILSVRC 2014, Szegedy et al. [32] developed GoogLeNet or Inception, that used convolutions of different sizes, reducing the number of parameters that need to be calculated. This network and its new versions are still widely used in medical image analysis.

A CNN can classify every pixel from the image if it receives patches collected around the pixel. The main inconvenience is that patches collected around neighbor pixels overlap and the same convolutions will be calculated many times. To overcome this disadvantage, the fully connected layers can be changed to convolutions and the CNN can, now, receive larger images than it was trained on no longer producing an output for a single pixel but a likelihood map. This new network is called fully Convolutional Neural Network (fCNN). The new drawback are the pooling layers which will create an output with less resolution, however Long et al. [33] solved this issue proposing a fCNN that replaces the pooling layers by up-sampling operators in order to increase the output resolution, and after that the high resolution features from the down-sampling path are stitched to the up-sampled outputs.

Ronneberger et al. [34] created U-Net, inspired in [33], a fCNN followed by an up-sampling path. There is the same amount of up-sampling and down-sampling layers but the correspondent down-sampling convolutional layers are connected to the up-sampling ones, concatenating features from both paths. A 3D U-Net was also created to deal with 3D data by Çiçek et al. [35], and Milletari et al. [36] built a 3D-variant network, called V-Net, that relies on 3D convolutional layers.

In medical image analysis the context is very important, for example, to detect some abnormalities, and the easiest way to increase this information is using larger patches as input. Moeskops et al. [37] developed a segmentation method for brain MRI that, using only one image, distinguishes the several tissues in that image relying on a network that uses multiple patches with different sizes and a different number of kernels in each layer obtaining spatial information. Their results are promising and reflect the robustness of the model. Also using brain MRI, Kamnitsas et al. [38] considered local and contextual

information using a network with two paths that processed images with multiple scales at the same time, obtaining top results in the challenges they participated.

DL made its first contributions in medical exams classification by making a prediction between 2 classes, usually healthy or not healthy. At the time, the number of available exams was small and transfer learning techniques started to become popular. Transfer learning uses a pre-trained network and its results to achieve better ones when using a new network to perform a similar task. It is possible to fine-tune a pre-trained network with medical images or use a pre-trained network to extract features, which carries the advantage of using the extracted features in already developed networks without the need to train new ones, however the best results occurred when Esteva et al. [39] tried to classify skin cancer lesions and Gulshan et al. [40] tried to detect retinopathies in retina images with both of them fine-tuning a pre-trained version of Inception v3. In conclusion, exams classification achieved good results when using pre-trained CNNs.

Organs and other medical image objects segmentation can lead to quantitative analysis of clinical parameters. This segmentation field is the one that presents the widest variety of methods going from segmentation based CNNs to Recurrent Neural Networks (RNNs), with RNNs being able to learn features from images sequences and time-varying data sets, accordingly to Salehinejad et al. [41], and starting to become more used in segmentation tasks. Still, the most used architecture in medical image segmentation is U-Net [34], not only but also, because this network can account for the whole image context.

More specifically, in cardiac imaging, the most common task is to segment the LV and the myocardium. Usually 3D and 4D (3D + time) data are analyzed slice by slice using 2D CNNs however Wolterink et al. [42] used 3D CNNs to quantify coronary artery calcifications. Some groups combine CNNs with RNNs such as Poudel et al. [43] that, from SA images, segmented LV using the U-Net architecture and then added RNN capability to remember informations from one slice when segmenting the next one, what improved the contours delineation near the apex of the heart.

Some challenges exist when it concerns cardiac segmentation, especially when segmenting the endocardium, which corresponds to the LV contour, and the epicardium. Endocardial segmentation is influenced by the presence of papillary muscles which present grey levels similar to the myocardium ones (they are both muscles) and epicardial segmentation is difficult because the myocardium around the right ventricle (RV) is very thin, comparing to the LV, and it is complicated to distinguish between this muscle and the surroundings. Complete segmentation of a cine CMR image sequence allows to evaluate LV volume variations as well as mass variations, which are of great interest to assess cardiac performance [44] and the degree of myocardial insufficiency.

The majority of the existent segmentation methods rely on the usage of SA images and on this note there are commercial softwares ready to delineate LV contours such as MASS (Medis, Leiden, The Netherlands), [45], and Argus (Siemens Medical Systems, Germany), [46], but these systems do not do it automatically, i.e. without the need of an operator. LA images are also used to segment the LV but in a less frequent manner: Hazirolan et al. [47] showed that there is a small difference in the obtained end-systolic volume, which is used to obtain EF, when using LA images versus SA images, with LA images providing the best results. Zhuang et al. [48] obtained good results when segmenting both 4 CH-LA and SA images using a Locally Affine Registration Method that created contours around the heart chambers which were later moved to the correct position using free deformations controlled by adaptive control points. Tsadok et al. [49] developed a method to segment both the LV endocardium and epicardium using 2 CH-LA and 4 CH-LA images, obtaining results comparable to expert manual segmentation, but, similarly to [48], did not use DL methods. These two later groups results and also [45] did not consider

the papillary muscles as belonging to the inside of the LV. Initially, Hautvast et al. [50] developed a method to automatically exclude papillary muscles from the LV blood pool and more recently Chuang et al. [51] considered the papillary muscles as belonging to the myocardial mass showing that LV mass and EF significantly increased and the EDV and ESV decreased, when comparing with the values obtained considering the papillary muscles as part of the LV blood pool.

More recently Arterys (Arterys, Inc., San Francisco, USA) showed up and developed a segmentation interface based on DL models that allows manual and semi-automated detection of both ventricles from cine CMR sequences [52]. Due to its novelty and to the best of my knowledge, is unknown if it considers the papillary muscles as belonging to the LV blood pool or to myocardial mass and if it works with LA images as good as it works with SA ones.

Considering all the achieved results and their limitations, one can conclude that there is still room for more research and for the development of new methods to segment LA cine CMR images using DL methods, which are becoming a powerful tool to analyze medical data. Therefore, these methods were used to segment different LA cine CMR images, in a supervised way, in order to quantify cardiac performance parameters and, consequently, facilitating the level of myocardial insufficiency evaluation.

2.12 Quantitative Myocardial Perfusion – Kinetic Model

Perfusion is defined as the oxygenated blood delivery to the different tissues of the body and this process occurs during systole which corresponds to the myocardial contraction forcing the blood to leave the ventricles. However, cardiac perfusion does not occur like this. Myocardial perfusion, i.e. perfusion of the myocardium, actually happens during diastole when the gas exchange can occur due to blood circulation in this muscle, since it is relaxed.

Myocardial perfusion imaging is used to diagnose CAD and detect any reduction in Myocardial Blood Flow (MBF) that occurs due to the narrowing of the arterial vessels. The evaluation of MBF is a good approach to determine regional perfusion deficits. Perfusion imaging is based on a dynamic contrast-enhanced acquisition which allows to detect the delivery of some contrast agent to the tissue being considered. To analyze the dynamic contrast-enhanced imaging data, kinetic models are used to quantify some parameters relevant to the final diagnosis [53].

Kinetic models translate the relation between the tracer, i.e. contrast agent, dynamics in the tissue and in the supplying artery. When it concerns the myocardium, the supplying signal should be measured as close as possible to the myocardial region under study but in practice it is measured in the LV in an attempt to account for variations in the rate, amount and delivery time of the contrast agent in the blood and it is known as Arterial Input Function (AIF), representing the concentration of tracer entering the tissue. Mathematically, in these models, there is a function that relates the dynamics at the input area with the ones at the relevant tissue and it is represented by the Impulse Response Function (IRF), as in equation 2.11.

$$C_{myocardium} = AIF(t) \otimes IRF(t) \qquad 2.11$$

A common model used to model these functions and quantify perfusion parameters is the Tofts Model (TM) [54]. This model, in its simpler version (Figure 2.12), quantifies two standardized quantities that characterize perfusion: the volume transfer constant, K^{trans}, and the Extravascular Extracellular Space (EES) fraction, v_e. The first measures the capillary permeability, i.e. the contrast agent flux rate from the blood to the EES and it differs between healthy subjects and patients, and the latter is the volume of EES present in a volume of tissue. From these two it is possible to define another quantity (equation 2.12), the rate constant, K_{ep}, that represents the flux rate between the EES and the blood and differs from K^{trans}. The TM can be implemented following equation 2.13.

$$K_{ep} = \frac{K^{trans}}{v_e} \tag{2.12}$$

$$IRF(t) = K^{trans} \times e^{-\frac{K^{trans}}{v_e}(t)} \tag{2.13}$$

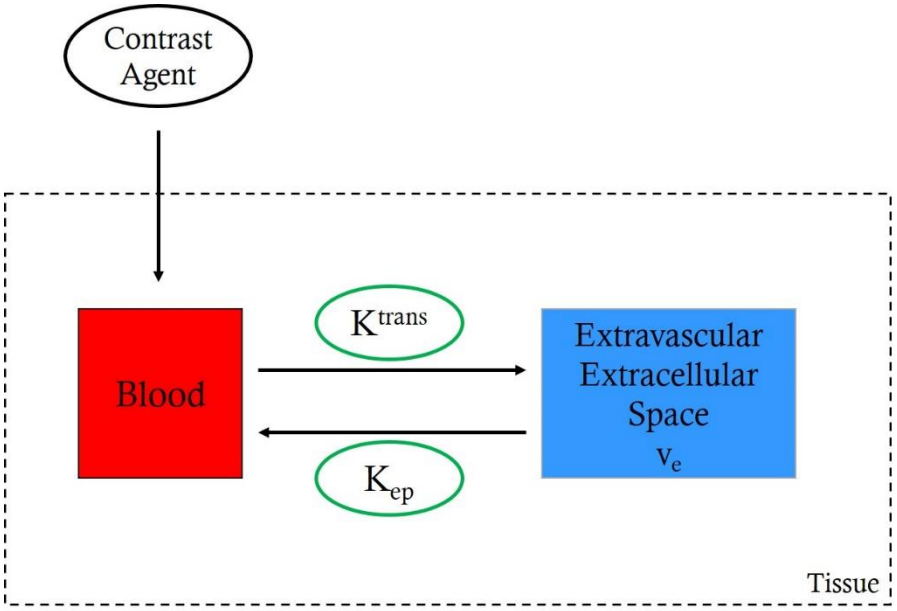


Figure 2.12 TM schematic representation. This simple representation shows the contrast agent intake into the blood, close to the tissue to study, and the consequent flow to the EES, modeled by K^{trans}. The rate constant, K_{ep}, represents the flux between the EES and the blood.

3 Methodology

The current chapter introduces the followed methodology to achieve the project's goals. It focuses on data processing, CNNs and these networks training procedure. It also distinguishes 2 methodologies: classification and regression.

3.1 Project Overview: Motivation and Goals

As mentioned in sections 2.1 and 2.11, CMR imaging is gaining terrain when it concerns the final diagnosis of CAD and a significant amount of research has been developed in the area of automatic segmentation of the most relevant structures present in the CMR images like the ventricular contours or the heart apex. The created contours are needed for quantitative analysis which is used to diagnose heart conditions. Completely automatic segmentation is challenging since the majority of all state of the art segmentation algorithms require manual corrections because a small number of segmentations contain imperfections that can lead to false diagnosis.

In the last decade, ML and, more specifically, DL algorithms hugely improved and the main focus of this project is to investigate the potential of DL algorithms to analyze CMR images through segmentation, registration and quantification of certain parameters mentioned before such as SV, CO and EF, facilitating the degree of myocardial insufficiency assessment.

Due to the broadness of this project it had to be carefully sectioned during its development. Primarily, the collected data underwent several operations in order to be correctly used by the NNs. During the project duration several networks were created and used because different data and goals were considered. Finally, a complete framework was implemented in order to fully automatically segment new data and produce the final evaluation, which was the quantification of some cardiac parameters.

Since there was some time left until the end of the project, new objectives were set. Using cardiac perfusion signals and regression networks, kinetic parameters were quantified in order to qualitatively classify the subject as healthy or not, i.e. with non-occluded coronary arteries versus occluded. To develop this extra part of the project new NNs were coded, since one changed from the classification to the regression scenario, and the signals had to be simulated using some *a priori* knowledge about them.

This section is divided in two parts: one concerning the initial and main part of the project, the classification scenario, and a second one dedicated to the regression part.

3.2 Classification Scenario

This section regards the initial and main part of the project, the classification scenario.

3.2.1 Data Processing

The collected data used during this project is made of (cine) functional CMR images, coming from a group of 55 patients, with 39 males and 16 females and ages between 12 and 92 years, collected along different imaging centers from Australia, Belgium, France, Germany, Japan, New Zealand, The Netherlands, United Kingdom and United States of America using different Philips (Best, The Netherlands) scans (models: Philips Ingenia and Philips Achieva) and sequences' parameters (Gradient Echo sequence with Flip Angles between 45° – 60°, Repetition Times between 2.4 ms – 3.7 ms and Echo Times between 1.2 ms – 1.8 ms) with magnetic field intensities of 1.5T and 3T. These functional images are short time sequences which show the heart motion and its different phases during the cardiac cycle, i.e. from a diastole to the next passing by a systole (Figure 3.1). The images were registered accordingly to the DICOM format and presented different sizes (224 x 224, 240 x 240, 256 x 256, 288 x 288, 320 x 320, 352 x 352, 384 x 384). Also, all the four different views mentioned before, 2 CH-LA, 3 CH-LA, 4 CH-LA and SA, were present in the data set however only the LA views were used for this project. The SA images could be used for comparative studies, but they were not present in a significative amount since only 2 of the 55 patients presented a cine sequence obtained with this view.

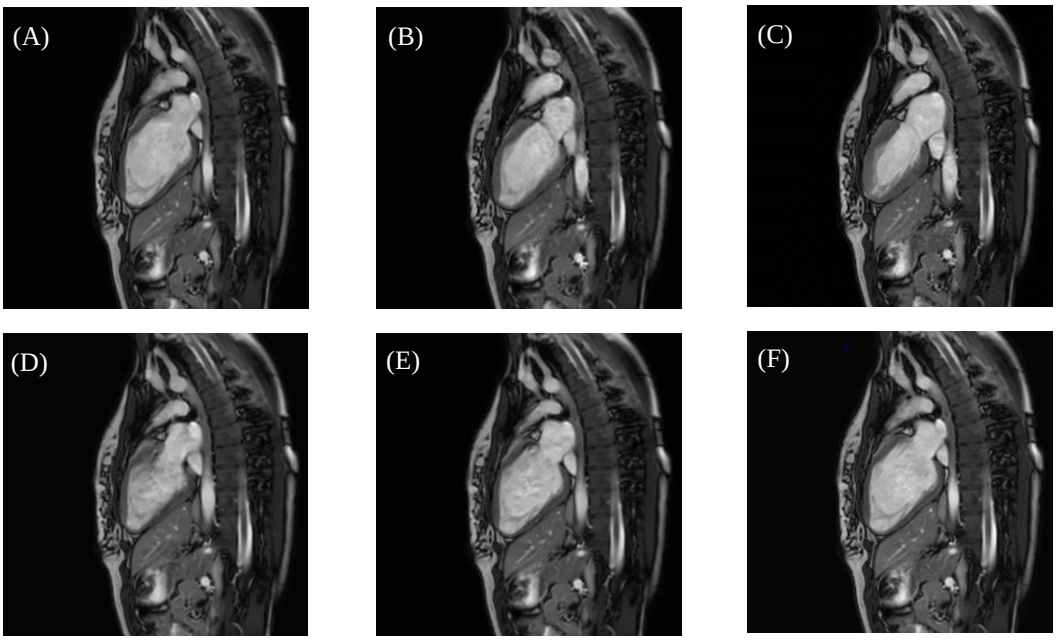


Figure 3.1 Functional (cine) CMR images. 2 CH-LA images representing six cardiac phases belonging to the cardiac cycle. (A) and (F) represent the most dilated phase of the myocardium (diastole), when the LV shows its maximum volume, and (C) represent the most contracted phase of the myocardium (systole), when the LV shows its minimum volume.

After selecting only the LA images from the 55 patients, 5 809 images were obtained and all of them were resized to the most common size present in the data set, 352 by 352 pixels, and normalized by changing the pixels' intensity range from 0 to 255, accordingly to the maximum intensity value of each image, to reduce the illumination differences in the data set. The images were split in 3 sets: training, validation and test. When developing a DL algorithm, the training phase is very important because is during this one that the model learns the features which will allow it to later analyze new images and segment them, therefore, the training data set must present some variability and be the largest of the 3 sets, corresponding to, approximately, 70% of all data. The validation set is made out of 20% and is used to fine tune the DL model parameters in order for it to provide the best results, and the test set is

made of the 10% left (Table 3.1). In Table 3.2 is shown the view variability among the images inside the training and validation sets.

Table 3.1 Information regarding the data set. All the 5 809 cine CMR images come from 55 patients and the data was split in training, validation and test sets accordingly to the percentages shown below. The number of patients and images corresponding to these percentages is also displayed.

55 Patients			
5 809 CMR Images			
352 x 352			
	Training Set	**Validation Set**	**Test Set**
Patients	38	12	5
Images	4 106	1 193	510
Percentages	70 %	20 %	10 %

Table 3.2 Information regarding the training and validation sets. Inside each of these data sets, the number of 2 CH-LA, 3 CH-LA and 4 CH-LA images varies accordingly to the displayed values. The training data set is made of 4 106 images and the validation set is made of 1 193 images.

Views (number of images)				
	2 CH-LA	**3 CH-LA**	**4 CH-LA**	**Total**
Training Set	1 574	1 038	1 494	4 106
Validation Set	379	407	407	1 193

3.2.2 Data Labeling and Augmentation

This project is inserted in a supervised learning scenario, therefore the images must be labeled so the network can learn without the user's influence. Two different types of labels were considered: distance maps and LV areas, and both of them were obtained based on LV contours.

Relying on anatomical knowledge and some expert tips, the contour of the LV was made. This contour was drawn using an interface previously developed in Matlab (Mathworks Inc, USA) made to interact with SA images, which was modified to work properly with LA images. To draw the contour there was no limit in the number of points used and on the length that goes from the first to the penultimate point it was not allowed to set any points (Figure 3.2), forcing the final point to always coincide with the first one. The developed interface is responsible to fit a spline, i.e. a function that approximates difficult curves using polynomials, to the points that the user considers belonging to the contour.

The contours were only drawn on the images belonging to the training and validation sets (there is no purpose in labeling the test set if we are building a DL algorithm to do so). When all contours were made, all of them were resampled to have 300 points and the center of mass of the LV was calculated. The center of mass was considered as the image center and was used to apply all future and needed

transformations to the image so the LV will always be present in the image. Once these final contours (5 299 contours) were drew and resampled, the images' labels were created.

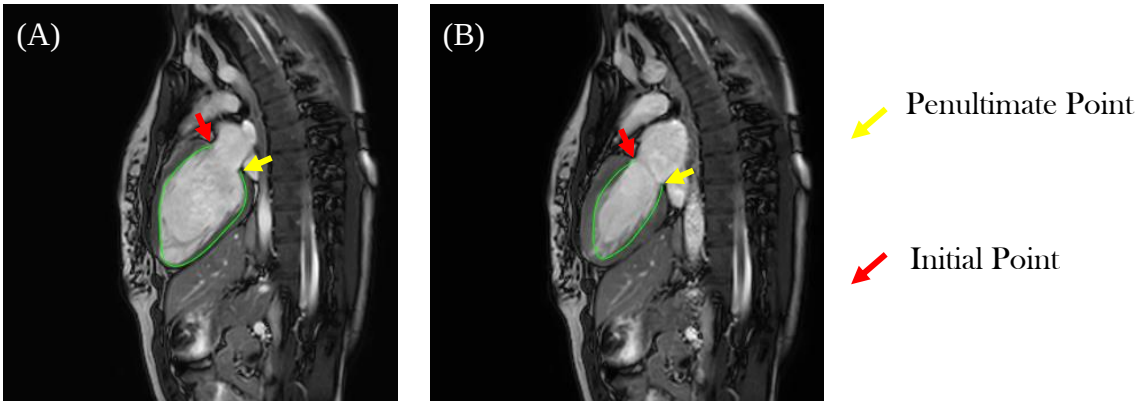


Figure 3.2 Two contours drawn using the Matlab interface. The number of points used to draw the contour in image (A) is greater than the one used in image (B). Between the initial and penultimate points no more points were allowed with exception to the final point which was placed over the initial point. The segment between the two points indicated by the arrows is not showed.

Distance maps quantify the distance between each pixel of the image and the pixels from some object of interest (usually something one wants to detect). In this case distance maps are used considering the Euclidean distance between each LA image pixel and the ground truth LV contour, previously drawn, and for this task the developed function finds the contour point closer to the image pixel considered at some instant, so it can get the final distance.

The other set of labels, the LV areas, was obtained also from the LV contours simply by closing and filling it. Both labels create only two classes for each image, i.e. LV/non-LV and LV contour/non-LV contour, however the distance maps are more interactive since the two classes depend on the choice of a threshold, returning a thinner or thicker contour line. Both label sets, distance maps and LV areas, were created using Matlab programming skills and were saved as matrices following the images' size, 352 x 352 pixels.

After labeling all images from training and validation sets, data augmentation was executed, only on the training set, because training a CNN with more data leads to better results since there are more examples. It is worth noticing that having many data does not necessarily lead to a better result, however, if the model is the best we can get, i.e. the parameters have the best values and the loss is the lowest, then having many data will lead to better results instead of overfitting.

Data augmentation was made using a Matlab function which produced random geometric transformations, such as rotation, translation and scaling, based on the previously obtained center of mass and following some constraints (Table 3.3) in order to produce images as real as possible. Rotations are measured in radians and the translations in pixels. From the original training images, 25 different transformations were created for each of these original images, resulting in a new training data set with 102 650 new CMR images. To obtain the corresponding distance maps and LV areas, the same transformations were also applied to the previously obtained label sets. Since the transformations can produce smaller images, when this happened the transformed images and LV areas were padded with zeros and the transformed distance maps were padded with the maximum value existent in the transformed distance map, until the 352 x 352 size was achieved.

Table 3.3 Transformations' constraints and correspondent values intervals used during data augmentation.

Transformations' constraints		
Rotation	**Translation**	**Scaling**
$\left[-\frac{\pi}{20}, \frac{\pi}{20}\right]$	[-10, 10]	[-0.3, 0.3]

3.2.3 U-Net: Implementation

After obtaining the final data sets correctly labeled, the training process started by establishing the CNN to use. From all the mentioned networks in section 2.11 and considering the project's goal, the selected one was U-Net [34].

U-Net (Figure 3.3) presents a contracting path responsible for capturing the image context and an expanding path responsible for providing precise location. Following the down-sampling path more filters are used to detect more specific features and the convolutions and poolings create even smaller feature maps each time, so in the end the whole image context is captured. The up-sampling path uses the feature maps from the contracting path, starting with the smaller ones, and evaluating the precise location of certain objects using the image context information, i.e. the high-resolution features from the down-sampling path. The cropping is carried out during the up-sampling path because, during the contracting path, there is a loss of the border pixels due to the convolutions and max-pooling, therefore, the final segmentation map only considers the pixels for which the context was completely captured.

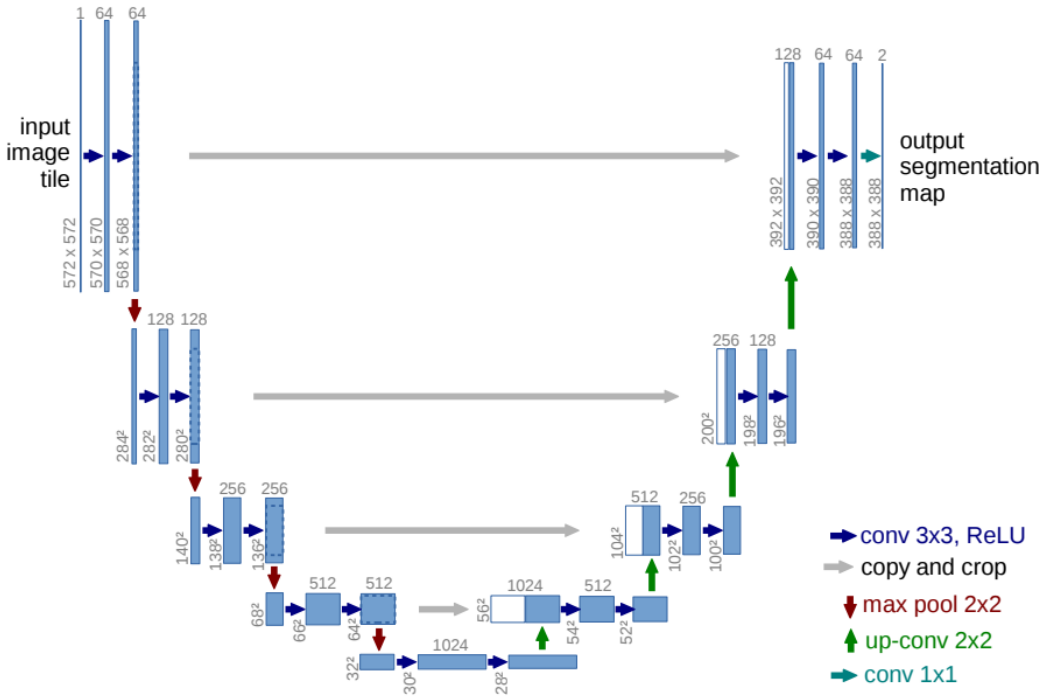


Figure 3.3 U-Net graphical representation. The blue rectangles represent feature maps, the input image and the feature maps' size is represented in the lower left corner and the number on the top of the rectangles represents the number of kernels (filters) used. The white rectangles in the up-sampling path represent the copied feature maps from the down-sampling path. Adapted from [34].

The U-Net implementation is available on-line and can be found in [55]. The used implementation in this project was adapted to work with the specified data and under this project's conditions not by changing the number or type of layers but by adding a dropout layer at the end of the down-sampling path and changing the number of filters and their size. The opensource implementation is based in a DL framework different from the one used in the project. The one used here is Theano [56], a Python library created to rapidly compute mathematical expressions since it was made to handle the NN's computational requirements used in DL, such as N-dimensional convolutions. Adding to Theano, Lasagne was also used. Lasagne [57] is a Python library as well but was specifically created to build and train NN's in Theano. Summarizing, an U-Net implementation was adapted to work on Theano and Lasagne (Figure 3.4) with the first being used to compute mathematical expressions, such as the convolutions and croppings, and the latest to implement the layers, such as convolutional, pooling, input and output. In this implementation softmax is used to compute each class probability and, therefore, cross-entropy is the loss function. Batch normalization is also performed.

```
net['input'] = InputLayer(shape=(None, 1, image_size, image_size), input_var=X)
net['contr_1_1'] = batch_norm(Conv2DLayer(net['input'], num_filters=feature_maps, filter_size=(3, 3),
     nonlinearity=lasagne.nonlinearities.rectify, pad='same', W=lasagne.init.GlorotUniform(gain='relu')))
net['contr_1_2'] = batch_norm(Conv2DLayer(net['contr_1_1'], num_filters=feature_maps, filter_size=(3, 3),
     nonlinearity=lasagne.nonlinearities.rectify, pad='same', W=lasagne.init.GlorotUniform(gain='relu')))
net['pool1'] = MaxPool2DLayer(net['contr_1_2'], pool_size=(2, 2), ignore_border=False)
```

Figure 3.4 Lasagne layers implementation. Using Lasagne library it is possible to easily define the input, convolutional, pooling and normalization layers, among others.

3.2.4 U-Net: Training

Even though there were two different types of labels, the used U-Net architecture was always the same. Accordingly to each label set, some details changed during the training process.

When considering the distance maps as labels, each distance map was loaded as a matrix which contained a wide range of values since they represent distances. To facilitate the prediction creation and the CNN output reading, one-hot encoding was performed. One-hot encoding is an operation that translates data values to zeros and ones. In the distance maps, a certain threshold was established and all matrix entrances whose value was superior to the threshold were translated to zeros and the others to ones, this way creating two classes: LV contour (1), if the distance map pixel value was smaller than the threshold, and non-LV contour (0) in the opposite case.

After this label encoding procedure, the images and the correspondent labels were loaded to start the training process (Figure 3.5 - A). Both components were saved as 4D tensors to facilitate the loading process following Theano configuration:

$$T = (B \times C \times H \times W) \quad 3.1$$

In this notation, B represents the minibatch size, C the number of classes, H the image height and W the width. The number of iterations, i.e. how many times, during training, a minibatch was created and, therefore, the training and validation loss was quantified, was also set so the training could start.

As already mentioned, the minibatch creation was random and to quantify the validation loss the 4D tensors were created from the validation set.

Table 3.4 shows the training parameters configuration used when considering the distance maps as labels. The parameter "Number of Feature Maps" was established accordingly to memory constraints and corresponds to its minimum number, the "Distance Map Threshold" was set that small because it represents the contour thickness and should be as precise as possible, the "Dropout Percentage, "Learning Rate" and "Minibatch Size" were optimized providing a small loss, and the "Number of Iterations" has two values to compare results in the following section. The "Image Tensor" second component, 1, means that the whole image was considered at the convolution time.

Table 3.4 Training Parameters Configuration. These parameters' values were used during training when the labels were distance maps and LV areas (in this latter case "Distance Map Threshold" was not considered).

Training Configuration	
Parameter	**Value**
Image Tensor	(5, 1, 352, 352)
Label Tensor	(5, 2, 352, 352)
Image Size	352 x 352 pixels
Number of Classes	2
Number of Feature Maps	16
Dropout Percentage	50 %
Minibatch Size	5
Number of Iterations	20 000 and 50 000
Learning Rate	0.001
Distance Map Threshold	2 pixels

Using the LV areas as labels (Figure 3.5 - B), these were loaded as matrices but since they already contained binary information there was no need to perform one-hot encoding. These labels were also saved as 4D tensors with the exact same size as the distance maps tensors, and the training configuration used here was the same for the training with distance maps (Table 3.4) with the exception that the parameter "Distance Map Threshold" was not necessary.

Apart from these 4 different trained networks, 3 more were trained to evaluate the influence of variability in the training set. This way, the difference was in this set, which only contained 2 CH-LA or 3CH-LA or 4CH-LA images, being all of them smaller than the original training set (Table 3.5), which counted with 102 650 images, and these networks were only used to classify correspondent images. The labels used by these 3 new networks were the LV areas and they were trained following the same training configuration presented before in Table 3.4, with the exception on the parameter "Number of Iterations" whose value was only 50 000. In total, 7 different networks were trained (Table 3.6).

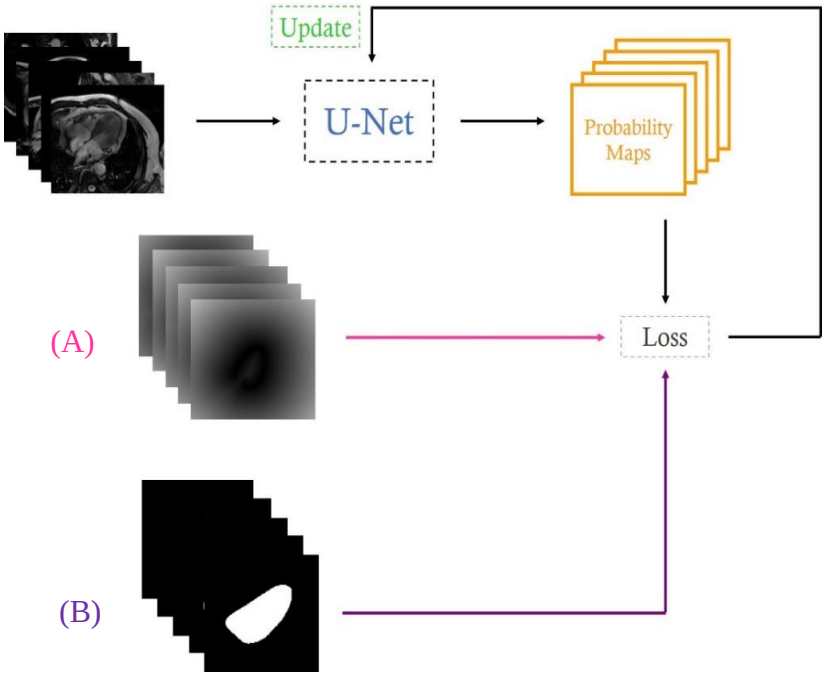


Figure 3.5 Training procedure using (A) distance maps and (B) LV areas as labels. For each case, A or B, and at each iteration, the images and labels are loaded as minibatches of 5, the U-Net produces a probability map, i.e. a prediction, for each image which is compared with the real label by calculating the loss (cross-entropy function). The loss is then used to update the CNN parameters, so the predictions improve at each iteration.

Table 3.5 Training and validation sets used by the three specific CNNs: the 2 CH-LA, 3 CH-LA and 4 CH-LA. Here are shown the number of images in each set.

Training sets (number of images)			
2 CH	**3 CH**	**4 CH**	**Original**
39 350	25 950	37 350	102 650
Validation sets (number of images)			
2 CH	**3 CH**	**4 CH**	**Original**
379	407	407	1 193

Table 3.6 Identification of all seven trained networks, used training sets and labels.

Trained Networks			
#	**Name**	**Training Set**	**Labels**
1	U-Net_20000	Original	Distance Maps
2	U-Net_50000	Original	Distance Maps
3	U-Net_FilledMasks_20000	Original	LV Areas
4	U-Net_FilledMasks_50000	Original	LV Areas
5	U-Net_2CH	2 CH	LV Areas
6	U-Net_3CH	3 CH	LV Areas
7	U-Net_4CH	4 CH	LV Areas

In image segmentation it is common to use the Dice Coefficient [58] as metric, since it makes a good comparison between the ground truth and the predicted segmentations. This coefficient can get values from 0 to 1, with 0 meaning that there is no spatial overlap between the ground truth and the prediction and 1 meaning a perfect overlap between the two structures, being the metric used in this project.

DL algorithms require a considerable amount of computational power and memory space to train, therefore the Medical Image Analysis Group GPUs were used for training. From the 4 Nvidia GPUs available, with ranging capacities from 8 to 12 GB, the Nvidia TITAN Xp with 8GB was the used one to train all networks mentioned in this project. Independently from the labels considered at the time, for the 50 000 iterations training session it took around 24 hours and for the 20 000 one it took around 10 hours. The training session included the images and labels loading, one-hot encoding (when needed) and the training and validation losses quantification.

3.2.5 Functional Parameters Quantification and User Independent Framework

After the CNNs training, their ability was used on the test set. The test set contained 510 images from 5 patients and, in order to obtain the functional parameters, all of them were subjected to several U-Nets actions to assess which ones work best, i.e. provide the best predictions, given the obtained parameters values.

As mentioned in section 2.2, to quantify the SV, EF and CO, the LV volume is needed and this quantity can be obtained from the predicted LV area, using the following equation:

$$\text{Volume} = \frac{8}{3\pi} \times \frac{\text{Area}^2}{\text{Length}} \tag{3.2}$$

Equation 3.2 represents the single-plane area-length technique [8], which approximates the LV shape to an ellipse and is used when only LA views are available. It uses the LV area and LA length to estimate the volume, specially in 2 CH-LA and 4 CH-LA images, however in this project it was also applied to 3 CH-LA views.

Considering the LV area predictions, the results of the U-Net_FilledMasks_50000 were used to quickly quantify the referred parameters since the training and validation errors were smaller, as it will be mentioned in the results section. From the area prediction, the area value was estimated counting the number of white pixels, i.e. pixels whose value was 1, and multiplying it by one pixel area. With the area value, in squared millimeters, the volume was calculated, converted to milliliters and the Ventricular volume – Time curve was plotted. Using the volumes, the functional parameters were calculated for each patient and accordingly to each considered view.

In order to compare the performance of the networks which used LV areas as labels, the SV, EF and CO were also quantified using the predictions created by the networks 5, 6 and 7. As previously mentioned network 5 was only used to predict the LV area from 2 CH-LA test images, network 6 from 3 CH-LA and network 7 from 4 CH-LA, from which the functional parameters were quantified and compared with the ones obtained from network 4 test predictions.

In equation 3.2 the LV LA length is one of the parameters which has to be obtained from the images. The straightforward method to get it would be by drawing a line from the heart apex to the valve points segment mid-point and measure it, however this procedure is user dependent, time consuming and it is difficult to mark the exact points (Figure 3.6). Therefore, in this project, a framework as user independent as possible was created in order to automatically segment the LV area, measure the LV LA and quantify the functional parameters.

It started with the creation of a new set of labels for the original data set. These labels were binary matrices that represented the three LV points used to measure the LA: the apex point (Figure 3.6 – orange point) and two valve points (Figure 3.6 – red points), obtained from the initially drawn LV contours. The valve points were the first and penultimate ones from the 300 points that define a contour (Figure 3.6 – green line) and the apex point was the further one from the line that connects both valve points (Figure 3.6 – yellow line). These new labels were created with data augmentation, similarly to the other two labels sets, containing 102 650 images, each one containing 2 valve points and 1 apex point, and were saved as 352 x 352 binary matrices.

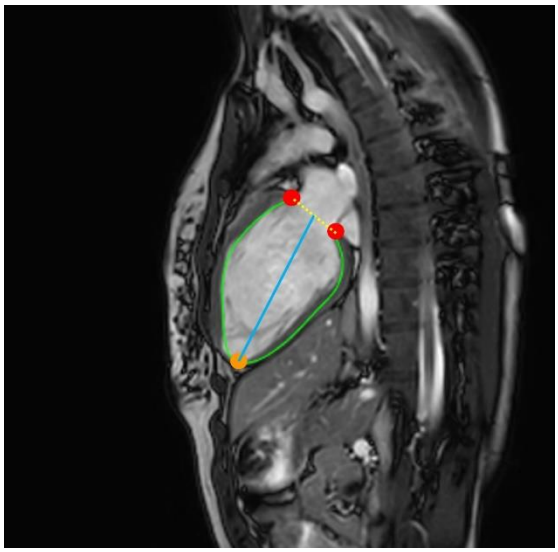


Figure 3.6 LV interest points and distances. The red points represent the valve points, which are the green contour first and penultimate points, and the orange point is the heart's apex, which is the green contour point further away from the intersection point between the yellow and blue lines. The dashed yellow line connects both valve points and the blue line connects the LV apex to the yellow line mid-point. The blue line represents the LA.

After the new training set labeling, a new CNN was trained maintaining the U-Net architecture. The training parameters configuration was quite different due to the labels 4D tensors, whose size changed since the number of classes augmented from 2 to 4: 2 valve points, apex point and background point, as shown in Table 3.7. This was the network number 8, whose training set was the original one and the labels were three points images.

Table 3.7 Training Parameters Configuration for the 8th trained U-Net, used for the user independent framework. These parameters' values were used during training when the labels were three points images.

Training Configuration – 8th U-Net	
Parameter	**Value**
Image Tensor	(5, 1, 352, 352)
Label Tensor	(5, 4, 352, 352)
Image Size	352 x 352 pixels
Number of Classes	4
Number of Feature Maps	16
Dropout Percentage	50 %
Minibatch Size	5
Number of Iterations	50 000
Learning Rate	0.001

Once trained, the complete test set was submitted to this new CNN action, creating a prediction for each image and the user independent framework started at this point. It is worth noticing that the CNN outputs a probability map, i.e. an image where each pixel value is a probability, therefore the pixels around the exact point location are not classified as background but have a smaller probability value than the real point. This characteristic led to a script creation, using Matlab, to receive the prediction and find the center of each point blob. With the obtained point coordinates, the lines shown in Figure 3.6 were drawn in order to measure the LV LA, eliminating the need for a user intervention. After this step and counting with the already predicted areas from networks 4, 5, 6 and 7, the final LV volume was obtained and the parameters quantified. Still regarding the Matlab script, it was view-dependent and, therefore, it was adapted to work with the three considered views in this project. The developed framework is represented in Figure 3.7.

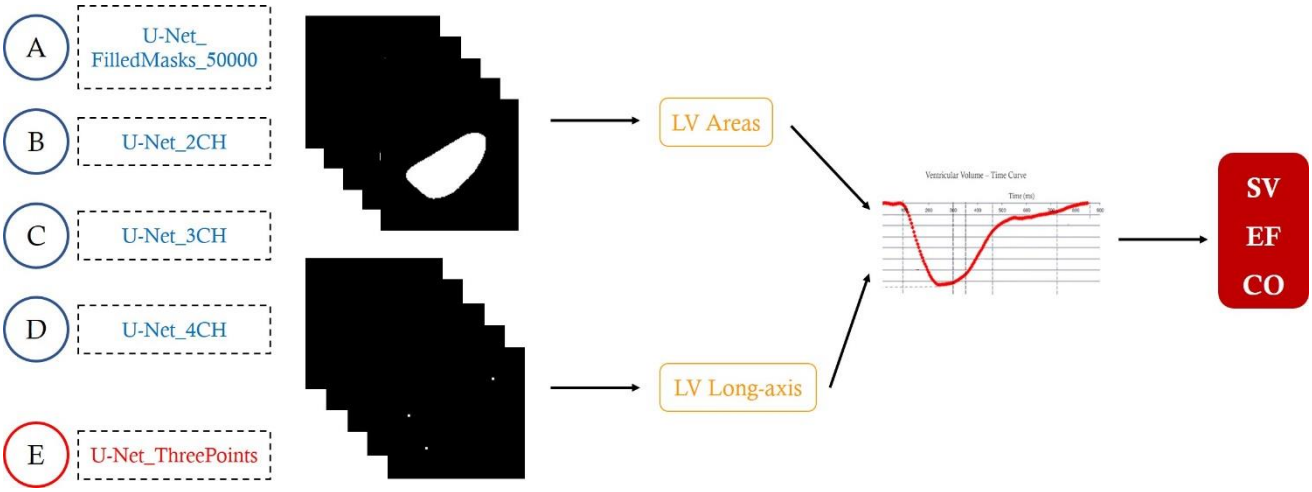


Figure 3.7 User independent framework. Each letter represents one trial, where the used network changed. The networks written in blue use LV areas as labels and the network written in red uses three points labels, from which the LV LA can be measured using the Matlab script. With the LV area and LA, the Ventricular volume – Time curve can be plotted and the functional parameters quantified.

3.3 Regression Scenario

This section presents the methodology followed during the regression scenario. During the regression part of this project, the goal was to classify the patients as healthy or non-healthy, accordingly to the obtained perfusion parameter K^{trans}. To quantify it, the TM was implemented as in equation 2.13, assuming a certain value for v_e and using simulated perfusion data.

3.3.1 Data Simulation

To be modeled, the TM requires the IRF(t) which can be obtained from the AIF(t) and the contrast agent concentration over time in the myocardium. To train the regression network, both the training and validation data were simulated accordingly to some prior knowledge about these functions.

The AIF can be modeled with a gamma function, since it reflects what happens with some concentration agent entering the tissue, i.e. a few moments after the tracer injection its concentration rapidly increases to the maximal value and then decreases as the body starts to eliminate it. Using Python language, this function was randomly created.

As the AIFs were created, it was necessary to label the data since we are under a supervised learning scenario. Since the goal was to quantify perfusion parameters such as K^{trans} and v_e, each signal label was these parameters values. Predicting two values is more complex than just one and since this part of the project was an extra, the v_e value was established *a priori* in 0.21, as in [59]. Also following the literature, a certain range of allowed values was established for K^{trans}, [0.01, 0.05], where the lowest values correspond to the occluded arteries scenarios.

Besides the AIF, the time signal representing the contrast agent concentration at the myocardium, $C_{myocardium}$, was also used. The tracer concentration takes more time to reach its maximum value in the tissue, and this maximal value is smaller than the AIF one, after which the concentration starts to decrease. From equation 2.9 it is possible to obtain the latter given the IRF(t). The IRF was obtained using the K^{trans} label value and the previously established v_e.

This way and to generate data, a Python script was created to return the AIF and the $C_{myocardium}$, saved as a 3D tensor and used as training data, and the K^{trans} value given v_e, saved as a row vector together with v_e and used as label for the generated signals. The validation set was also generated following this methodology. Both the training and the validation sets were generated at training time to calculate the losses but were not saved due to memory constraints.

3.3.2 Regression Network

Regression networks are simpler than classification ones, since they are less deep, i.e. have less layers, they predict values and not probability maps, even though they still rely on convolutions. In this project a regression network was created containing only 6 convolutional layers with batch normalization, 3 pooling layers, dropout and fully connected layers. The used loss function was the MSE.

With the network established, it was trained accordingly to the following configuration (Table 3.8). The 3D tensor used to store the signals contains 2 signals, AIF and $C_{myocardium}$, which is the minibatch size, extracts 1 feature from the signals, the K^{trans}, and the signals size is 150 points. The row vector used to store the labels contains the 2 parameters, K^{trans} and v_e, predicted for each signal (v_e was not predicted but was added to the vector). Less iterations and a higher learning rate were considered due to the fact that regression networks work with less dimensional data and are simpler, leading to a training session time of around 9h, using the same GPUs used to train the classification networks. To evaluate this regression network predictions, i.e. the constant K^{trans}, the RMSE and R^2 were calculated at validation time. R^2 measures how close the predictions are to the optimal fitting line by returning the variability percentage of the response variable explained by the fitting model, i.e. if this variable value reaches 100%, it means that the predicted values always coincide with the original ones with all observations matching the optimal fitting line.

Table 3.8 Training Parameters Configuration for the regression network. These parameters' values were used during training when the label was K^{trans}.

Training Configuration – Regression Network	
Parameter	**Value**
Signals Tensor	(2, 1, 150)
Label Tensor	(1, 2)
v_e	21 %
Number of Feature Maps	16
Dropout Percentage	40 %
Minibatch Size	2
Number of Iterations	20 000
Learning Rate	0.1

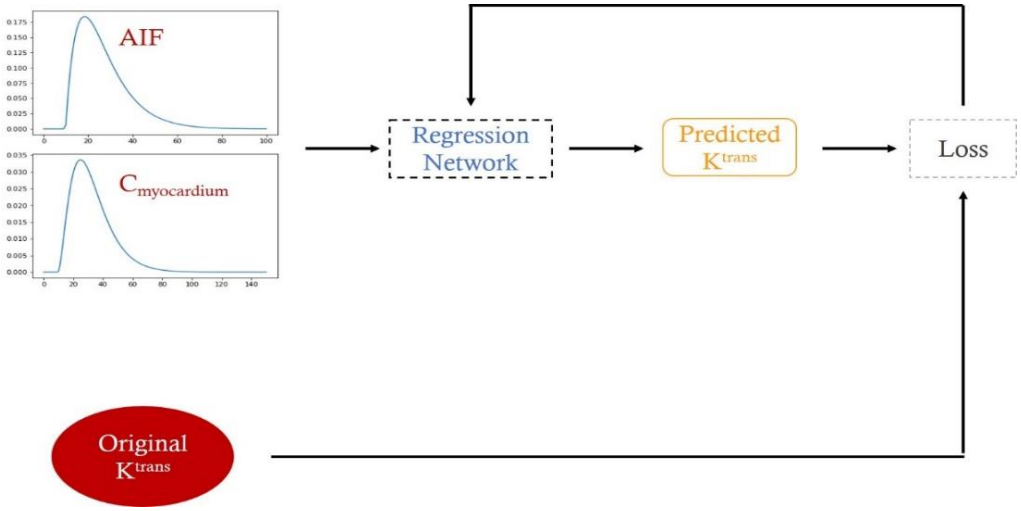


Figure 3.8 Regression network training procedure. Using 2 signals as input, AIF and $C_{myocardium}$, the regression network made a prediction for the K^{trans} which was compared with the original value of this parameter and the loss was quantified in order to update the network parameters.

4 Results

The current chapter regards all the data processing, labeling, CNNs' training and predictions, parameters quantification and regression results obtained during the project duration.

4.1 Data Processing and Labeling

All the results obtained from operations performed on the images such as contouring, contour resampling, label creation and data augmentation are shown in this section.

The following images (Figure 4.1) show the drawn contours before being resampled, i.e. the initial segmentation step. One resampled example, i.e. the final segmentation step, is presented in Figure 4.3 together with the LV center of mass used to apply the transformations.

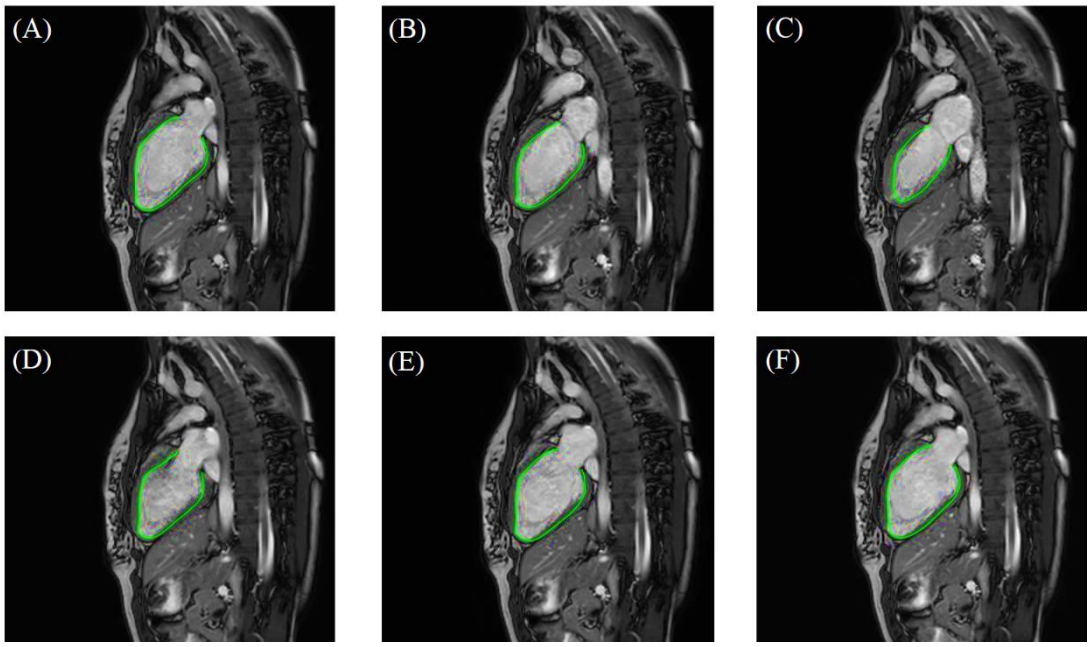


Figure 4.1 Functional CMR images with respective contours. Each image represents one cardiac cycle phase with the respective contour drawn in green without restrictions in the number of used points. These images show the first step towards the final training and validation images segmentation.

The obtained distance maps and LV areas labels are presented next. In Figure 4.2 (A) it is possible to visualize that the closer to the contour a point is, the darker it appears in the distance map, and in Figure 4.2 (B) one can see the original contour when a proper threshold is selected. Figure 4.4 shows the respective LV area label for the presented cardiac cycle phase.

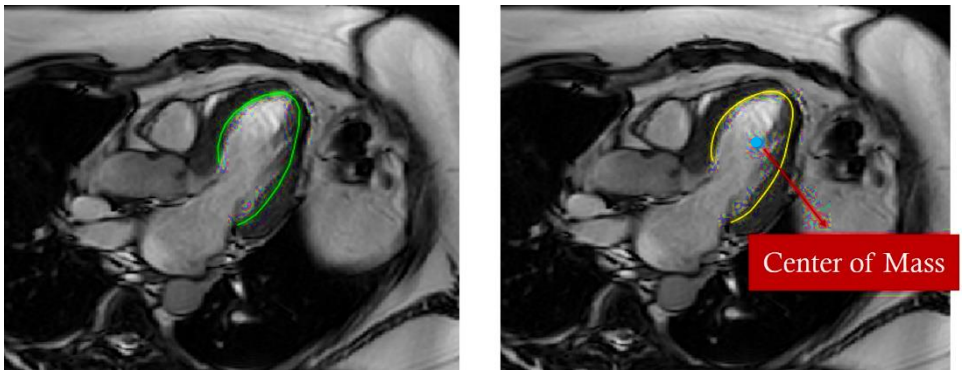


Figure 4.3 Contour resampling and center of mass. The image on the left shows the initially drawn contour and the image on the right shows the same contour but resampled to 300 points (smoother contour) and the blue point represents the contour center of mass where all the performed transformations were applied.

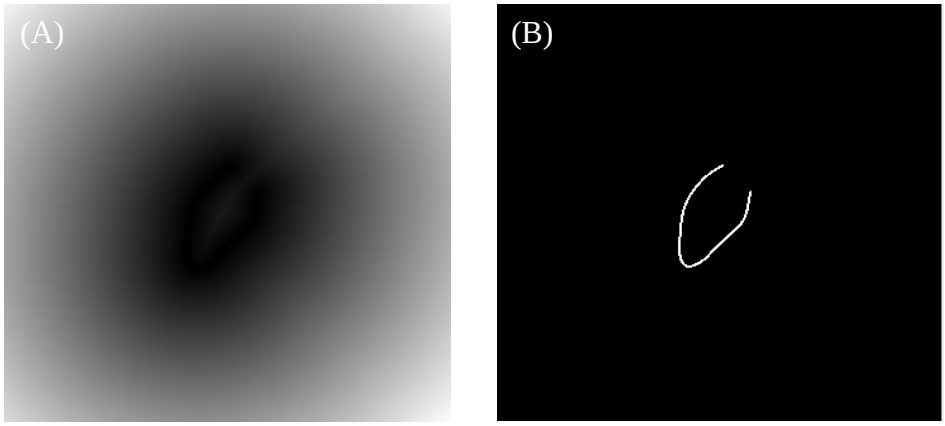


Figure 4.2 Distance maps labels. (A) Distance map corresponding to the image in Figure 4.1 (B). Pixels closer to the contour are colored in black and the further ones have grey shades. (B) Pixels in white correspond to the contour in Figure 4.1 (B) and the distances are calculated based on this contour.

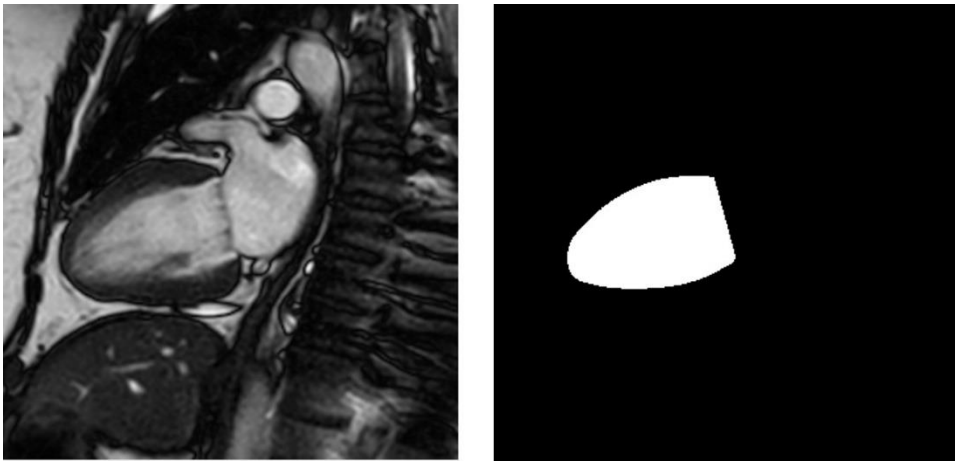


Figure 4.4 LV area label. The image on the right shows the binary matrix that corresponds to the image on the left label. It was obtained by closing the contour and filling the space enclosed by this structure.

To train the 8th mentioned U-Net, the one used to predict the apex and the valve points, the labels had to be created from the LV contours as well. Using the previously created LV contour, the first and penultimate points were selected to represent the valve points, and the apex point was the contour point further away from the valve points segment mid-point, as shown in Figure 4.5.

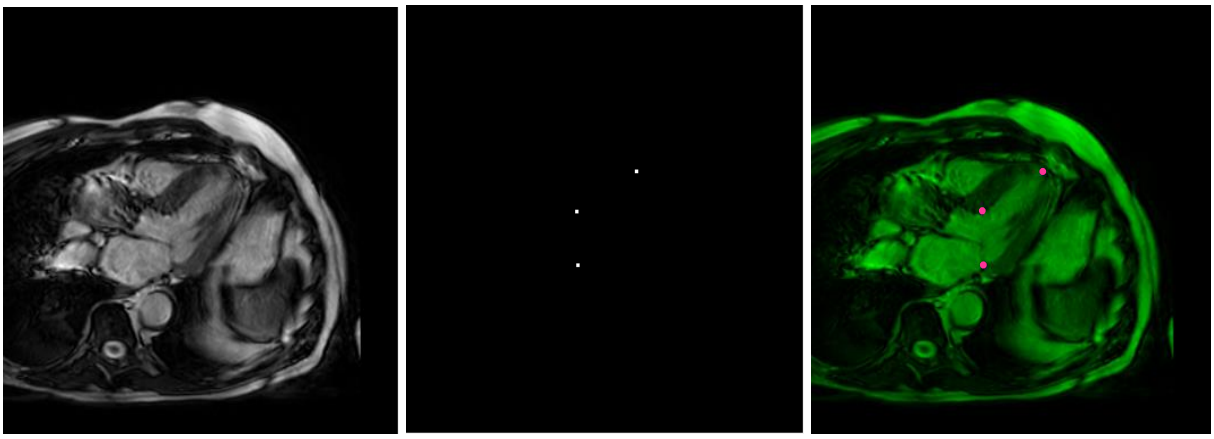


Figure 4.5 Three points label. The middle image shows the binary matrix with the apex and valve points correspondent to the LV shown on the image on the left. The image on the right is the original image (left) and respective label (middle) overlap.

4.2 U-Nets Training and Predictions

8 U-Nets were trained considering different training configurations. The first networks to be trained were, accordingly to Table 3.6, U-Net_20000 and U-Net_50000, with both networks being trained over the original training set and considering the distance maps as labels. Following the training configuration presented in Table 3.4, the obtained losses, for both networks, are shown next (Figure 4.6), together with a prediction result for one randomly selected validation image, from each network (Figure 4.7).

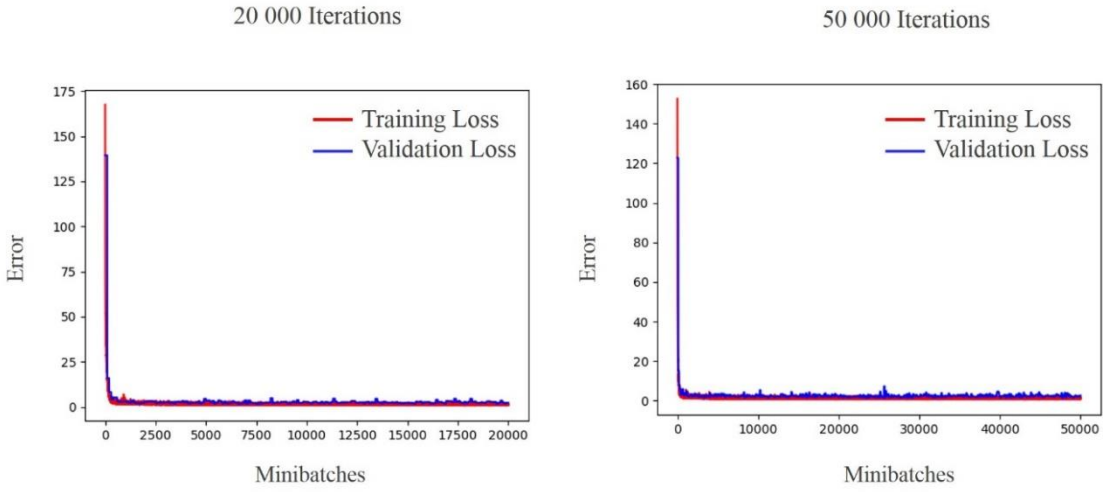


Figure 4.6 Training (red curve) and validation (blue curve) errors for both networks, U-Net_20000 and U-Net_50000. On the left are represented the losses for the network trained during 20 000 iterations, U-Net_20000, and on the right are the losses for the network trained during 50 000 iterations, U-Net_50000.

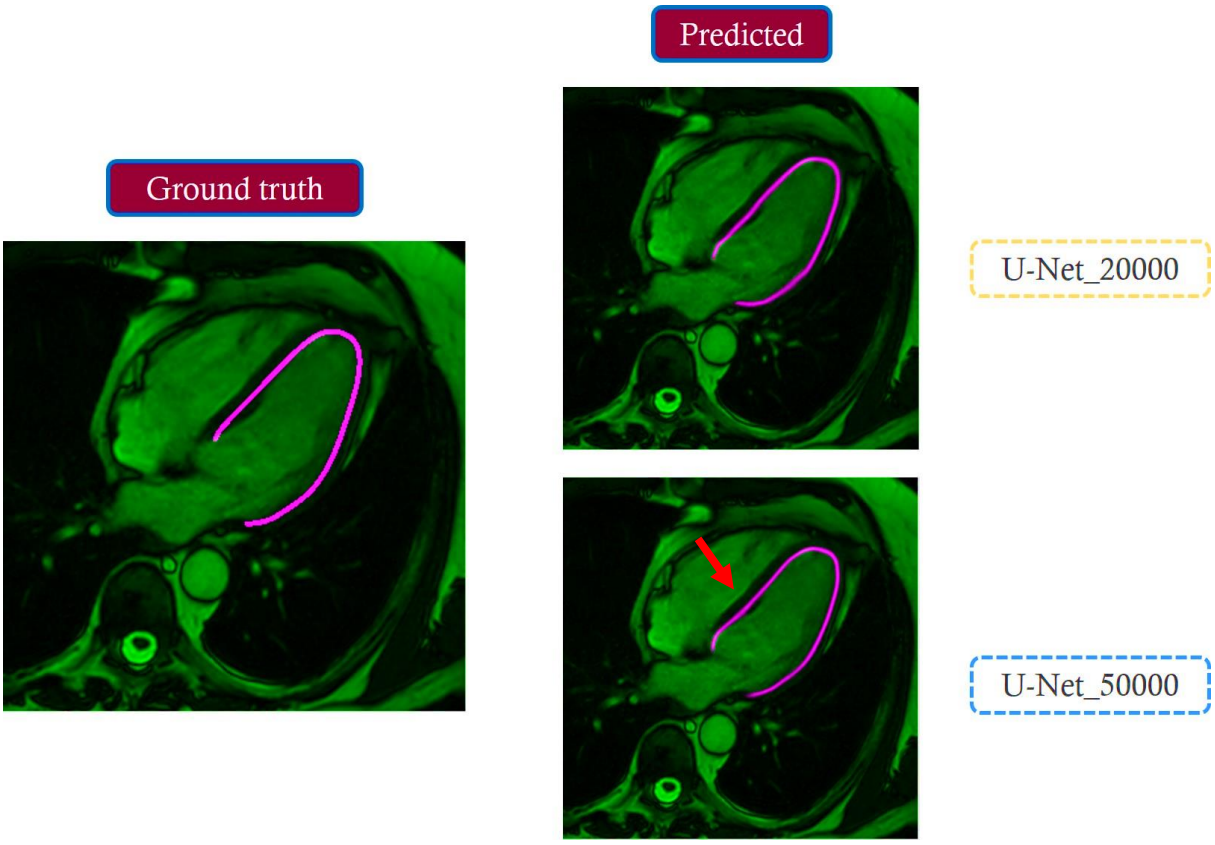


Figure 4.7 LV contour predictions obtained from both networks U-Net_20000 and U-Net_50000. On the left it is represented a randomly selected validation set image with the respective contour overlapped, i.e. the ground truth. On the right are represented the LV contour predictions created by each of the considered networks. The red arrow points to an area where the LV contour seems to adapt better to the LV.

The evolution of the Dice Score during training and validation for U-Net_20000 and U-Net_50000 networks is represented in Figures 4.8 and 4.9 and, to facilitate its reading, in Table 4.1 the mean Dice Coefficients for these networks' validation predictions and the correspondent Standard Deviation (SD) are presented.

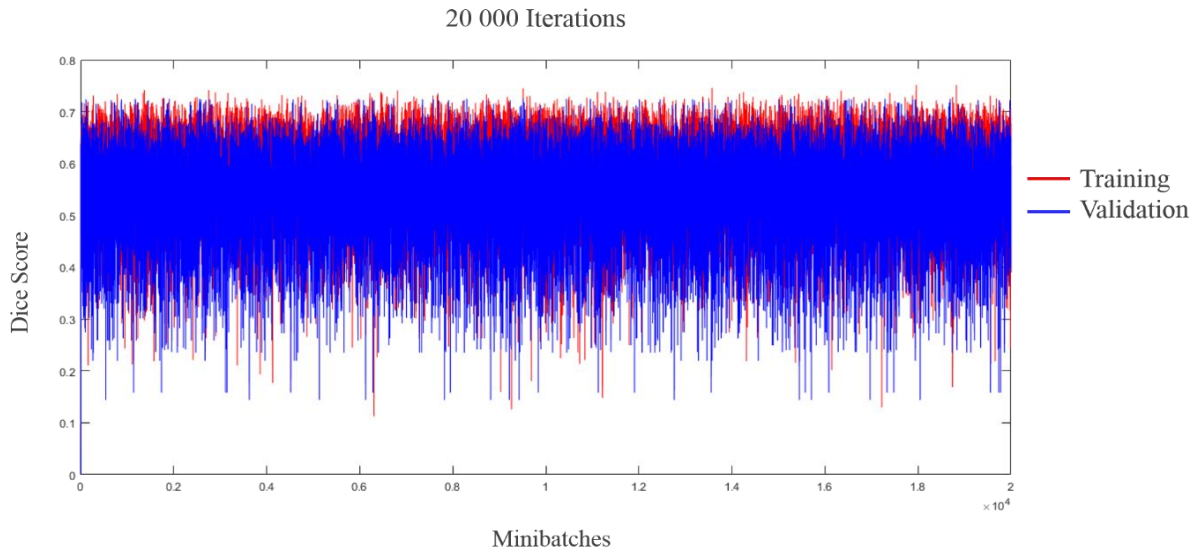


Figure 4.8 Dice score evolution during training (red) and validation (blue) for U-Net_20000.

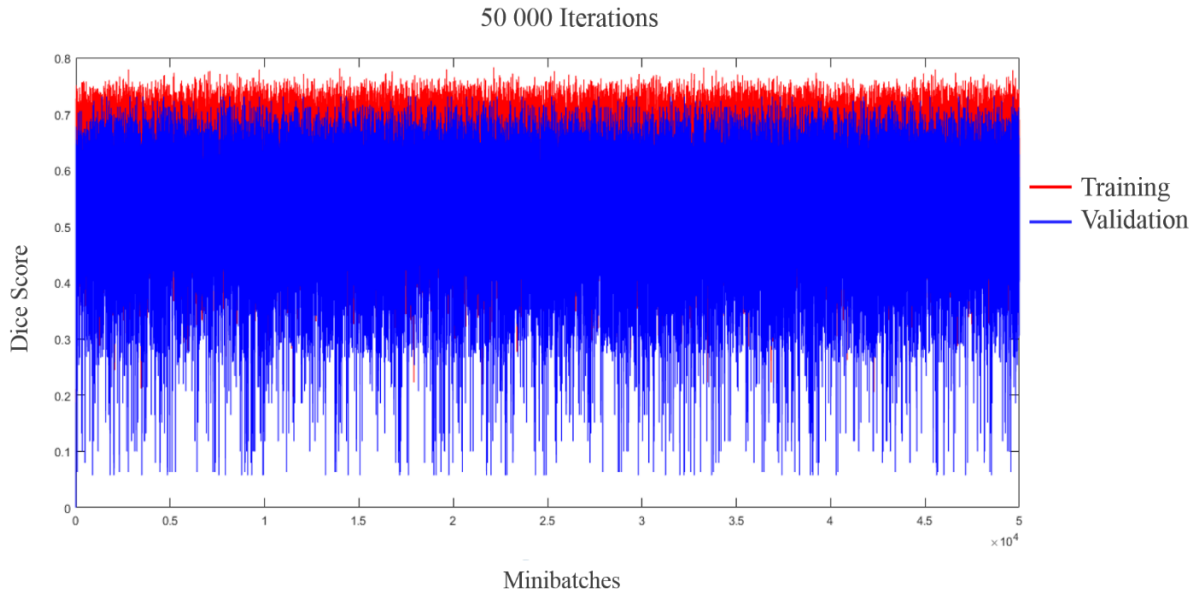


Figure 4.9 Dice score evolution during training (red) and validation (blue) for U-Net_50000.

Table 4.1 Mean Dice Coefficients and SD for both U-Net_20000 and U-Net_50000 networks' validation predictions.

Dice ± SD – Validation	
U-Net_20000	**U-Net_50000**
0.535 ± 0.093	0.531 ± 0.102

The next set of results are relative to the networks trained using the original data set but using the LV areas as labels, the U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000. Here, the training and validation errors for each network (Figure 4.10), the predicted labels for a randomly selected validation image (Figure 4.11), the Dice score evolution during training and validation for both these networks (Figures 4.12 and 4.13) and the mean Dice Coefficients (Table 4.2) obtained during validation are presented, similarly to what was realized with the two previous networks.

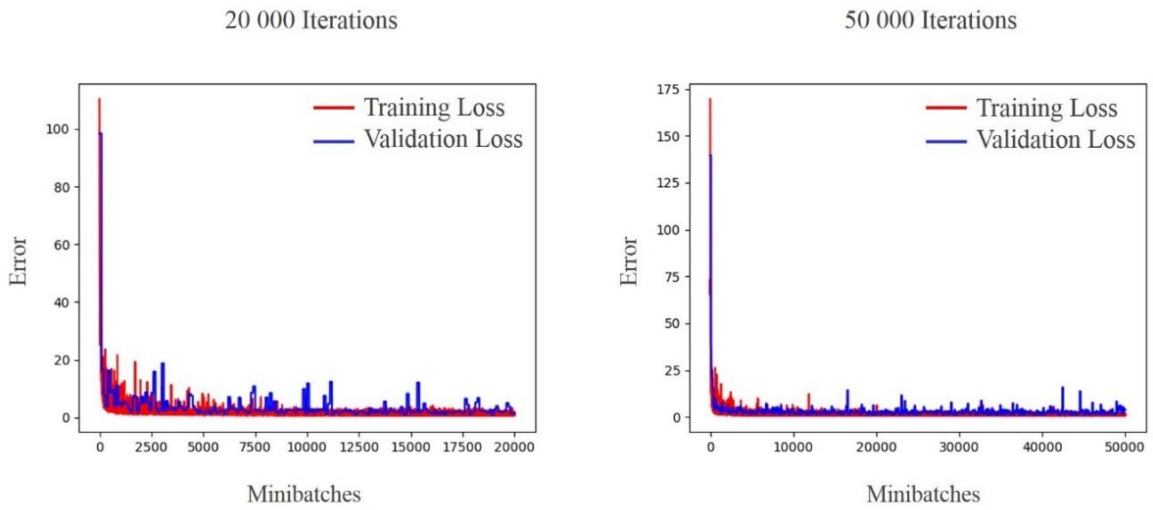


Figure 4.10 Training (red curve) and validation (blue curve) errors for both networks, U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000. On the left are represented the losses for the network trained during 20 000 iterations, U-Net_FilledMasks_20000, and on the right are the losses for the network trained during 50 000 iterations, U-Net_FilledMasks_50000.

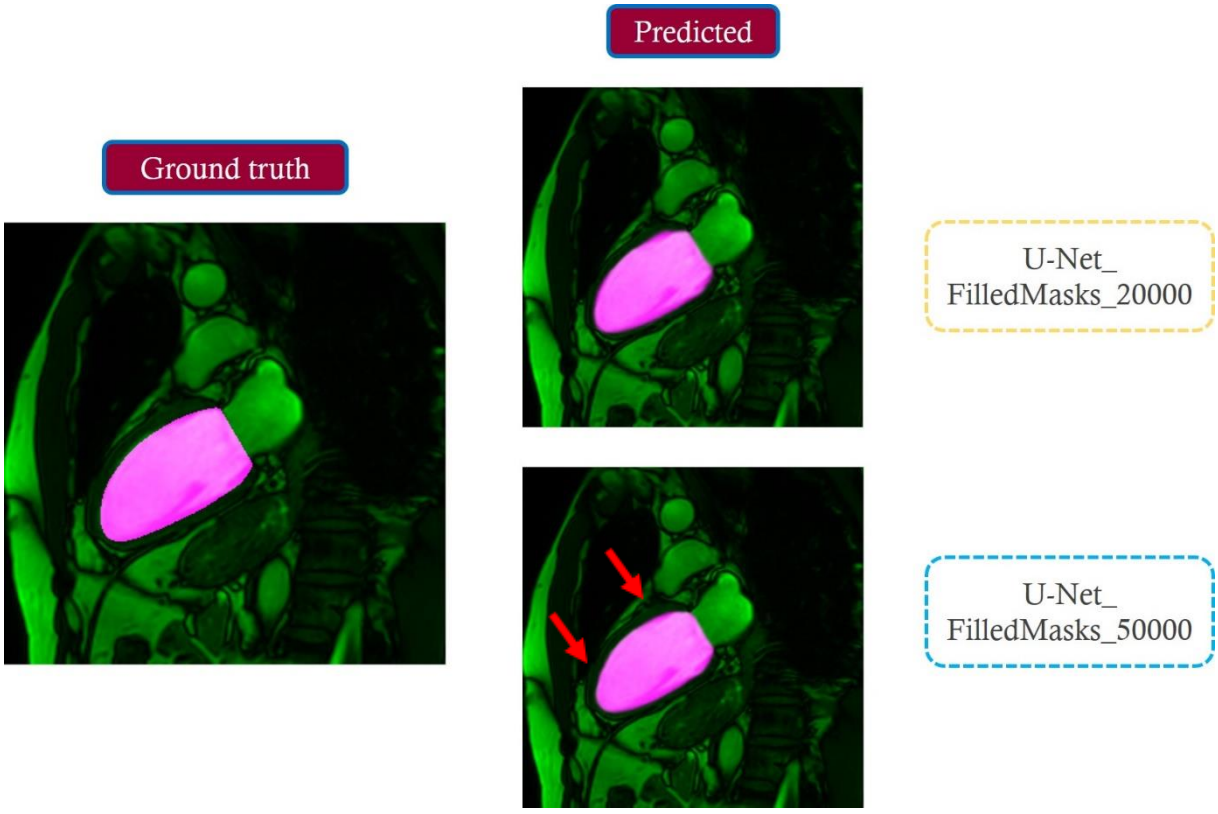


Figure 4.11 LV area predictions obtained from both networks U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000. On the left it is represented a randomly selected validation set image with the respective area overlapped, i.e. the ground truth. On the right are represented the LV area predictions created by each of the considered networks. The red arrows point to areas where the LV area prediction seems to fit better comparing to the U-Net_FilledMasks_20000 prediction.

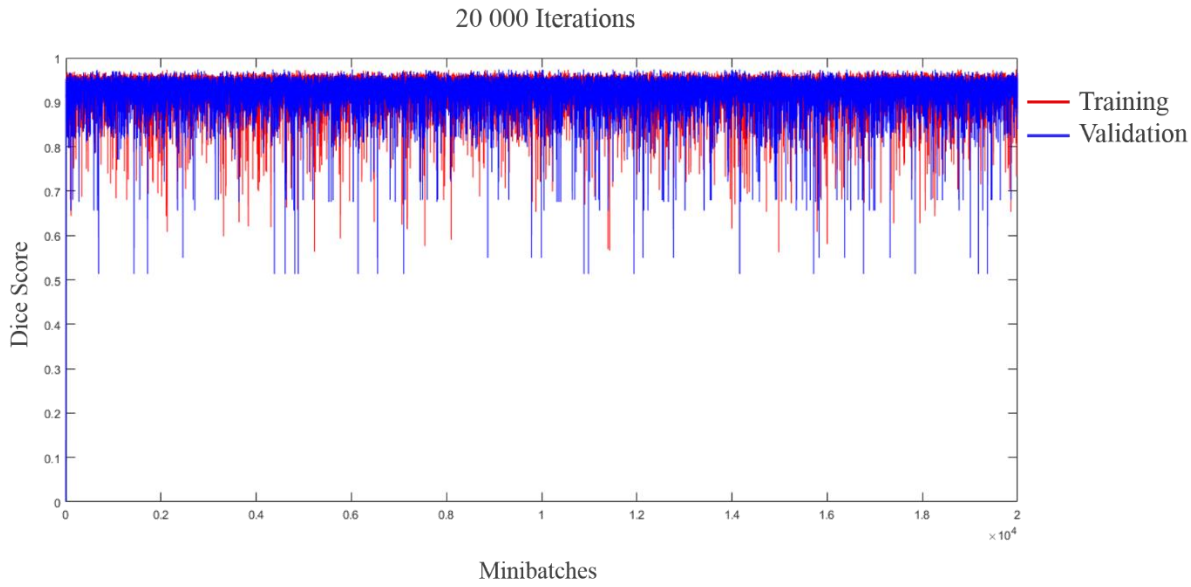


Figure 4.12 Dice score evolution during training (red) and validation (blue) for U-Net_FilledMasks_20000.

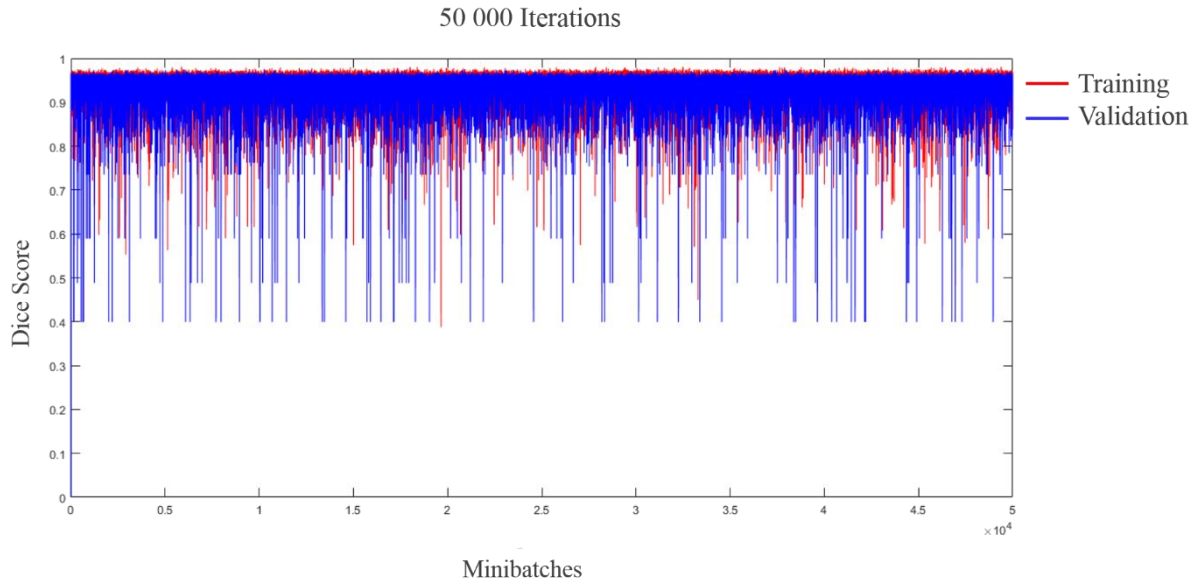


Figure 4.13 Dice score evolution during training (red) and validation (blue) for U-Net_FilledMasks_50000.

Table 4.2 Mean Dice Coefficients and SD for both U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000 networks' validation predictions.

Dice ± SD – Validation	
U-Net_FilledMasks_20000	**U-Net_FilledMasks_50000**
0.924 ± 0.037	0.935 ± 0.037

The following set of U-Net results corresponds to the U-Net_2CH, U-Net_3CH and U-Net_4CH networks. All these networks had, as labels, LV areas however all of them were trained based on different training sets with each of them using a data set only containing 2 CH-LA, 3 CH-LA or 4 CH-LA images, respectively.

Since the training and validation errors graphs were very similar between the three networks (leading to the same conclusions), only the graph correspondent to the U-Net_3CH is presented in Figure 4.14.

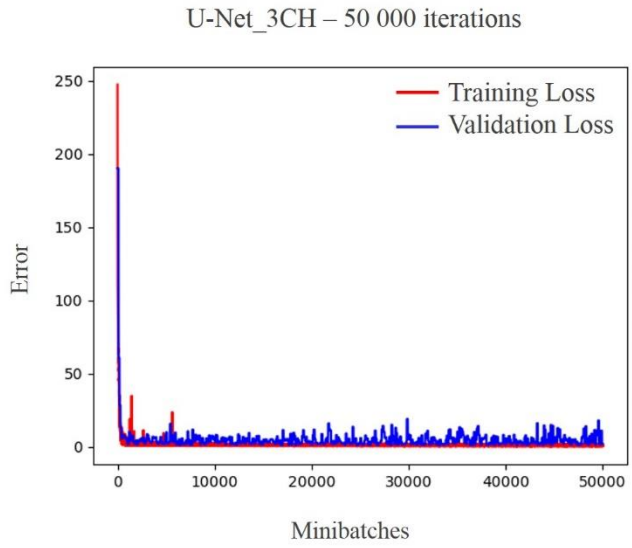


Figure 4.14 Training (red curve) and validation (blue curve) errors for the U-Net_3CH network. This network was only trained for 50 000 iterations. The graphs obtained for U-Net_2CH and U-Net_4CH were similar to this one.

Figure 4.15 shows, for each network, the LV area prediction for a randomly selected validation image accordingly to the considered view and in Table 4.3 the respective mean Dice Coefficients obtained from the validation predictions are presented. Similarly to what was presented before, Figure 4.16 show the Dice score evolution during network training and validation for the three networks.

Table 4.3 Mean Dice Coefficients and SD for U-Net_2CH, U-Net_3CH and U-Net_4CH networks' validation predictions.

Dice ± SD – Validation		
U-Net_2CH	**U-Net_3CH**	**U-Net_4CH**
0.941 ± 0.022	0.908 ± 0.049	0.931 ± 0.028

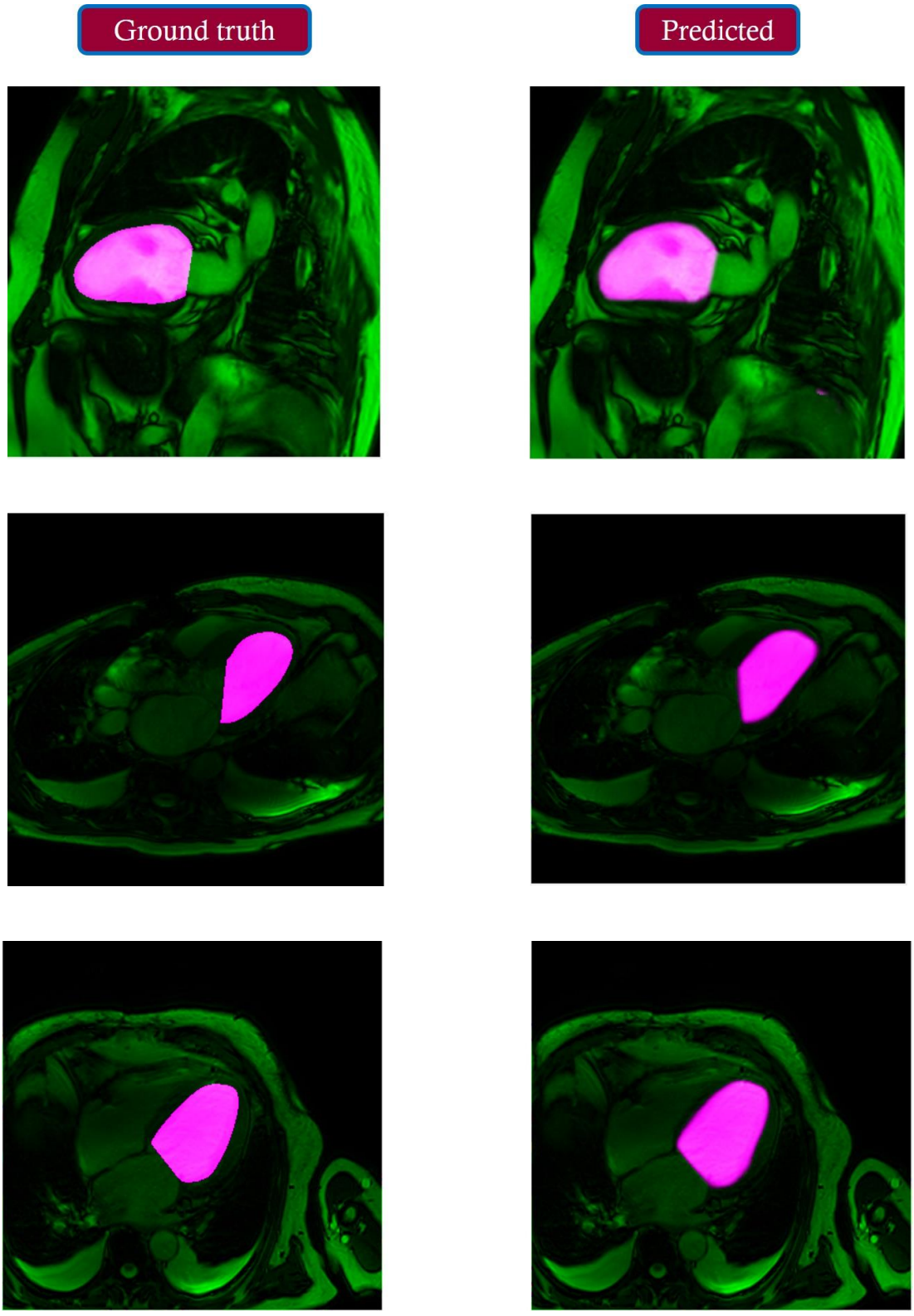


Figure 4.15 LV area predictions obtained from three networks. The first line of images represents the results from the network specifically trained to classify 2 CH-LA images, the U-Net_2CH, in the second line are the results from the network trained with 3 CH-LA images, U-Net_3CH, and in the bottom line there are the results from U-Net_4CH, trained with 4 CH-LA images. The column on the left shows the ground truth and the one on the right the network prediction.

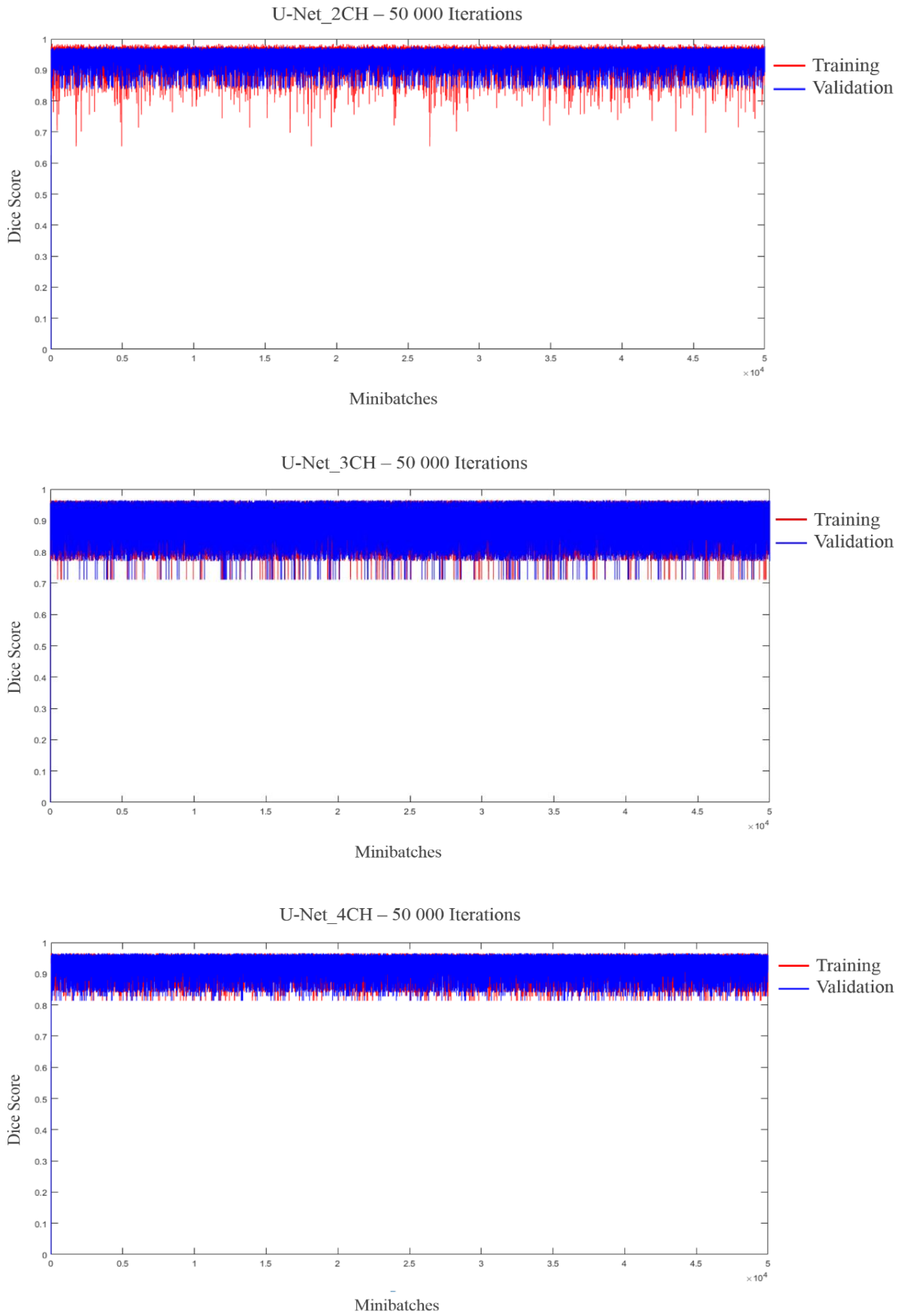


Figure 4.16 Dice score evolution during training (red) and validation (blue) for U-Net_2CH, U-Net_3CH and U-Net_4CH.

To finalize the results section regarding the U-Nets, the results from the 8th U-Net, the one that predicts the apex and valve points are presented next. As mentioned before, the prediction returns blobs, i.e. an area around the predicted point, instead of just a single point (Figure 4.17), and to fight this issue a script was developed to find the middle point of each blob.

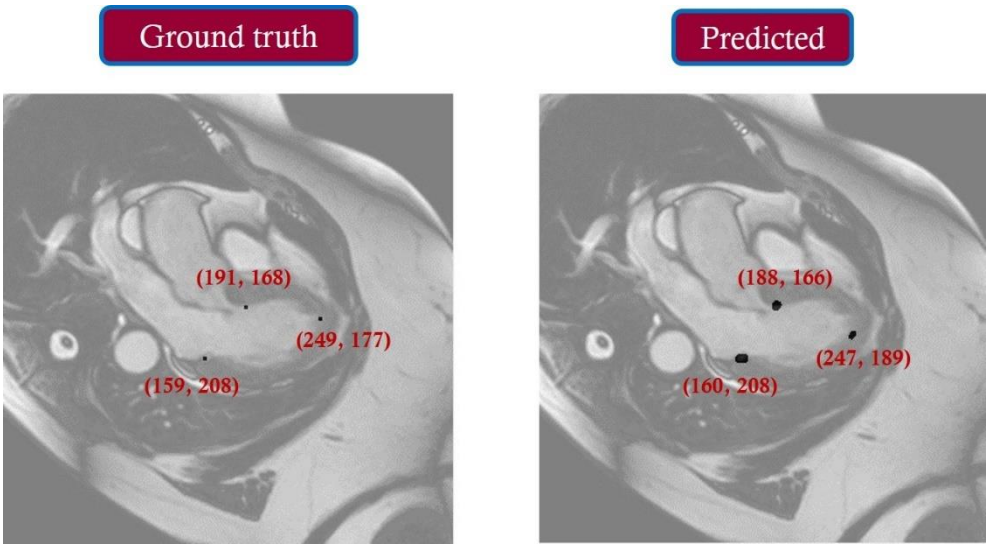


Figure 4.17 Apex and valve points coordinates prediction obtained from the 8th trained network. This network was trained with the original data set and the used labels were binary matrices representing the three considered points.

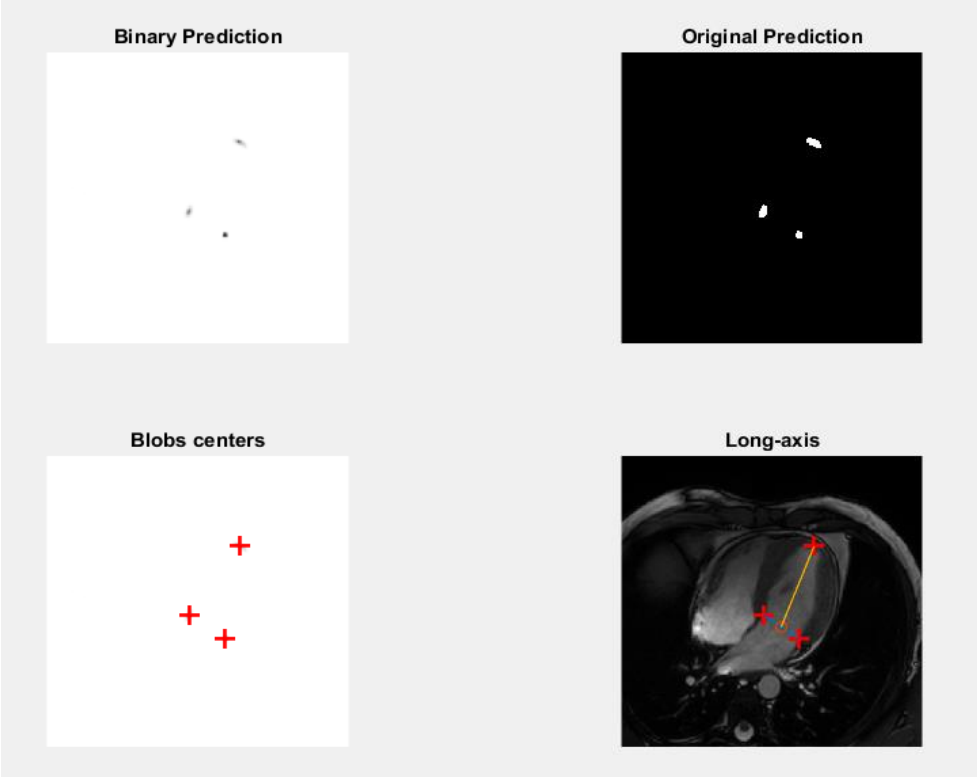


Figure 4.18 Matlab script output. For a given CMR image, the network number 8 predicts the apex and the valve points returning 3 blobs (top right corner image), from which each blob center is calculated (bottom left corner image). Having each point coordinates a line is drawn between both valve points and the LV LA is obtained connecting the apex point to the previous described line mid-point (bottom right corner image).

Figure 4.18 shows the output of this mentioned script, considering one random validation image. Besides getting the precise location of the predicted points, the script returns to the user the length of the LV LA (yellow line), with which the ventricular volume and functional parameters can be quantified.

Given that this 8th network predicts 3 different points, a mean Dice Coefficient was calculated for each (Table 4.4) from the validation predictions (Figures 4.19, 4.20 and 4.21).

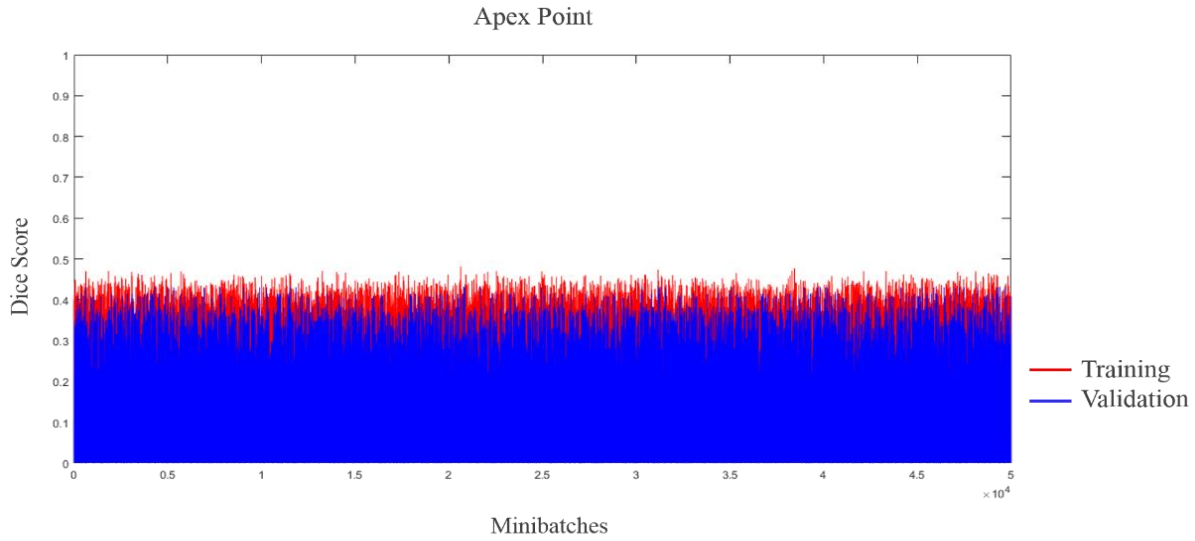


Figure 4.19 Apex point Dice score evolution during training (red) and validation (blue) from the 8th trained U-Net.

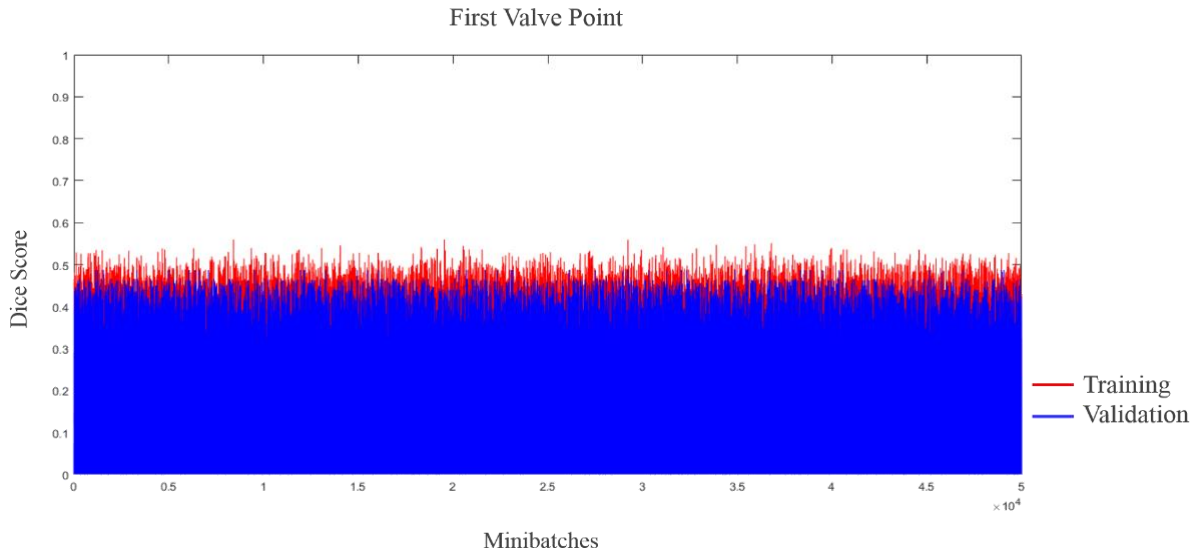


Figure 4.20 First valve point Dice score evolution during training (red) and validation (blue) from the 8th trained U-Net.

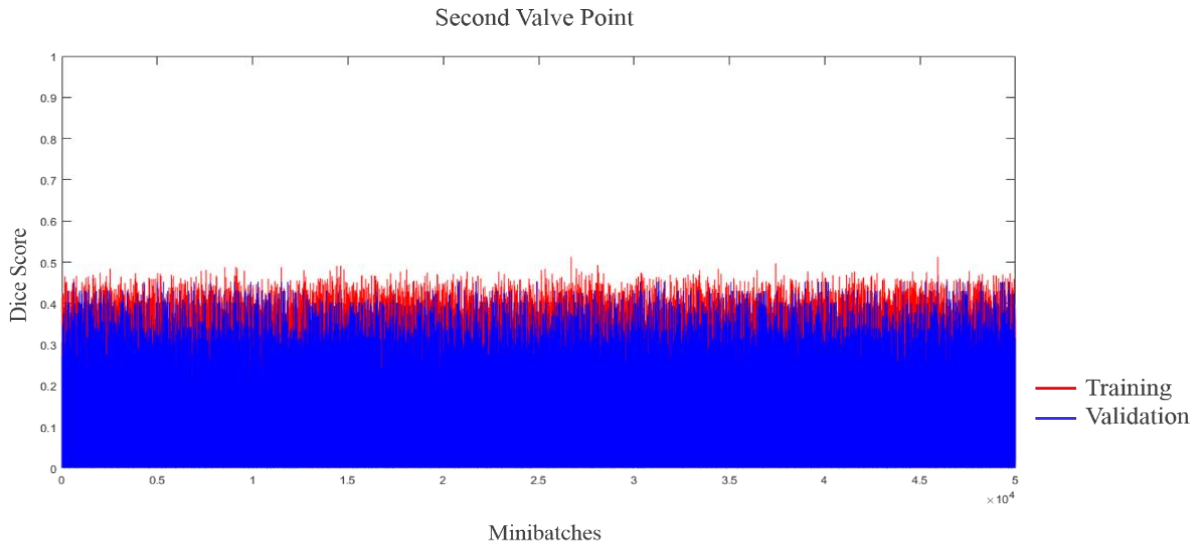


Figure 4.21 Second valve point Dice score evolution during training (red) and validation (blue) from the 8th trained U-Net.

Table 4.4 Mean Dice Coefficients and SD for U-Net_3points network, considering the 3 existent classes: apex point, first and second valve points.

Dice ± SD – Validation		
U-Net_3points		
Apex point	**First valve point**	**Second valve point**
0.079 ± 0.106	0.154 ± 0.136	0.097 ± 0.106

4.3 Ventricular volume – Time curves And Functional Parameters

The test set (Table 4.5) was used to obtain these next results. This set contained 5 patients and 510 images including the three types of views. As explained, the CMR images were functional, i.e. each cardiac cycle phase could be individually visualized, and the Ventricular volume – Time curves were calculated for each cardiac cycle. The cardiac cycle could be described by 30 or 40 phases, depending on the patient since the data came from different scanners and using different acquisition protocols. In total there were 14 cardiac cycles: 5 with 4 CH-LA images, 5 with 2 CH-LA and 4 with 3 CH-LA images.

To obtain the Ventricular volume – Time curves, the LV area and the three points predictions were used. The following results regarding this curve were obtained from the U-Net_FilledMasks_50000, U-Net_2CH, U-Net_3CH and U-Net_4CH LV areas and the 8th U-Net points predictions. A curve was plotted for each cardiac cycle. In Figures 4.22, 4.23 and 4.24 it is possible to compare the resultant curves from each network for a certain test patient, accordingly to each view.

Table 4.5 Test set characteristics. Among the 5 test patients, there were 14 cardiac cycles with a varying number of phases and different views.

Test set Characteristics		
Patient	**Cardiac Cycle Phases**	**View**
# 1	1 – 40	4 CH-LA
	41 – 80	2 CH-LA
	81 – 120	3 CH-LA
# 2	1 – 30	4 CH-LA
	31 – 60	3 CH-LA
	61 – 90	2 CH-LA
# 3	1 – 40	4 CH-LA
	41 – 80	3 CH-LA
	81 – 120	2 CH-LA
# 4	1 – 40	4 CH-LA
	41 – 80	2 CH-LA
	81 – 120	3 CH-LA
# 5	1 – 40	2 CH-LA
	41 – 80	4 CH-LA

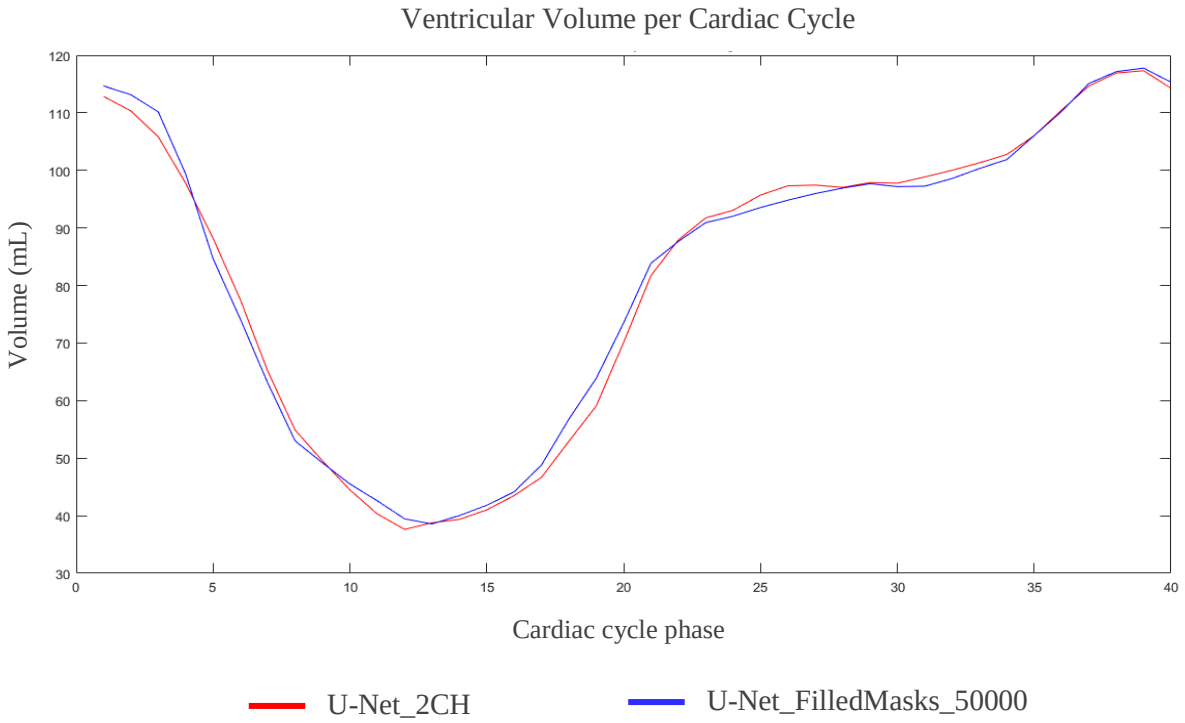


Figure 4.22 Ventricular volume – Time curves obtained using LV area predictions from two different networks: U-Net_2CH and U-Net_FilledMasks_50000, for one of the 5 cardiac cycles containing 2 CH-LA images, randomly selected.

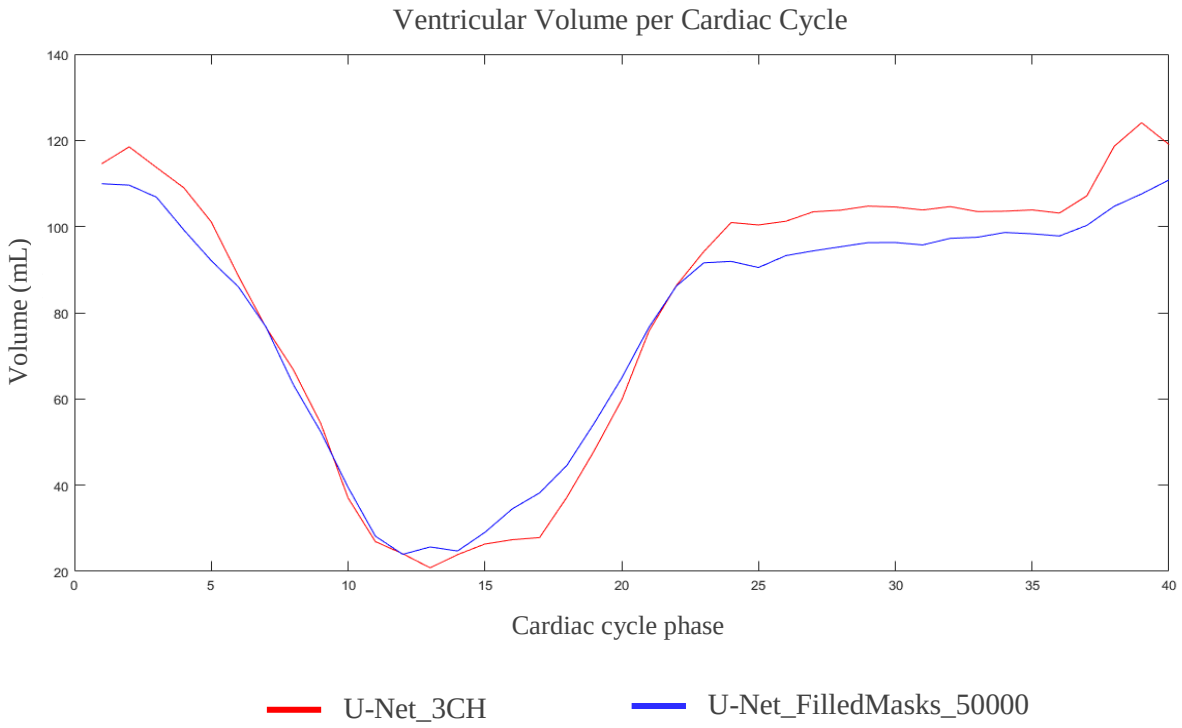


Figure 4.23 Ventricular volume – Time curves obtained using LV area predictions from two different networks: U-Net_3CH and U-Net_FilledMasks_50000, for one of the 4 cardiac cycles containing 3 CH-LA images, randomly selected.

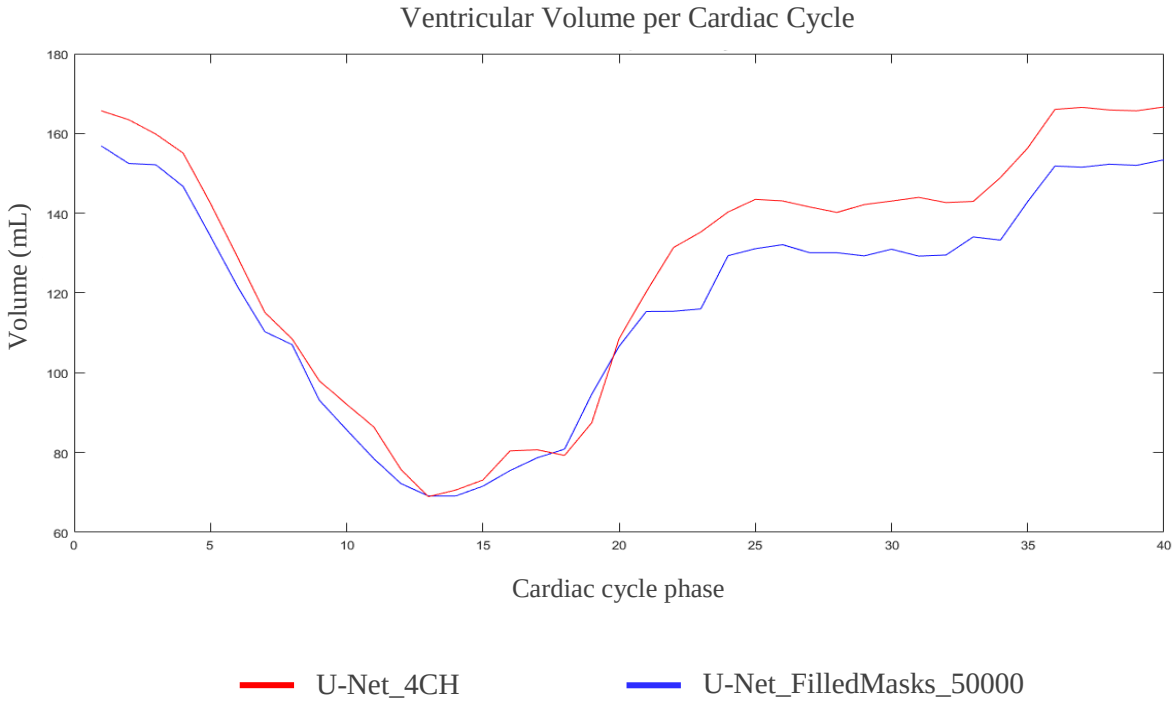


Figure 4.24 Ventricular volume – Time curves obtained using LV area predictions from two different networks: U-Net_4CH and U-Net_FilledMasks_50000, for one of the 5 cardiac cycles containing 4 CH-LA images, randomly selected.

To help deciding which of the three previous networks' pairs (U-Net_FilledMasks_50000 *vs* U-Net_2CH, U-Net_FilledMasks_50000 *vs* U-Net_3CH and U-Net_FilledMasks_50000 *vs* U-Net_4CH) provided the best LV area predictions, paired t-tests were performed. The following table presents its p-value results, considering a significance level, α, equal to 0.05.

Table 4.6 Paired t-tests results for LV areas comparing each pair of networks: U-Net_FilledMasks_50000 vs U-Net_2CH, U-Net_FilledMasks_50000 vs U-Net_3CH and U-Net_FilledMasks_50000 vs U-Net_4CH.

Paired t-test – P-value ($\alpha = 0.05$)		
U-Net_FilledMasks_50000 *vs* U-Net_2CH	**U-Net_FilledMasks_50000 *vs* U-Net_3CH**	**U-Net_FilledMasks_50000 *vs* U-Net_4CH**
0.968	0.760	0.354

With the obtained volumes, and using equations 2.1, 2.2 and 2.3 the functional parameters were quantified. In order to evaluate how good the parameters obtained for the test patients were, reference values were obtained from the training set. Table 4.7 shows the obtained reference values and standard deviation for the SV (mL), EF (%) and CO (L/min). In the SV and EF cases, the accepted ranges for each parameter were established as Reference Value ± 1 SD. Since the HR was unknown for all patients and in order to obtain the reference value for the CO, a reference value from literature for HR was used (Table 2.1) together with the previously calculated SV from the training subjects.

Table 4.7 Reference values and SDs for SV, EF, HR and CO used to analyze this project's results. The used HR reference value comes from literature since it was unknown for all training patients.

Parameter	**Reference Value (± SD)**
SV	95.42 mL (± 33.80 mL)
EF	60.7 % (± 17.4 %)
HR	65 beats/min
CO	6.2 L/min (± 2.20 L/min)

The following table shows the obtained values for the SV, EF and CO for each of the 14 test cardiac cycles and makes the distinction between the values obtained from LV area predictions coming from the U-Net_FilledMasks_50000 and the ones from the U-Net_2CH, U-Net_3CH and U-Net_4CH.

Table 4.8 Functional parameters obtained for each of the 14 cardiac cycles, using LV area predictions resultant from 4 different networks. The SV is presented in mL, EF in % and CO in L/min.

		Functional Parameters											
		U-Net_FilledMasks_50000			U-Net_2CH			U-Net_3CH			U-Net_4CH		
Cardiac Cycle		**SV**	**EF**	**CO**	**SV**	**EF**	**CO**	**SV**	**EF**	**CO**	**SV**	**EF**	**CO**
# 1	1 – 40	75.0	61.5	4.9	----------			----------			124.4	77.8	8.1
	41 – 80	117.8	86.8	7.7	113.8	88.2	7.4	----------			----------		
	81 – 120	85.9	78.2	5.6	----------			93.8	81.8	6.1	----------		
# 2	1 – 30	64.2	49.2	4.2	----------			----------			46.7	39.7	3.0
	31 – 60	40.1	52.1	2.6	----------			41.1	52.8	2.7	----------		
	61 – 90	38.6	41.8	2.5	25.2	31.7	1.6	----------			----------		
# 3	1 – 40	84.1	63.1	5.5	----------			----------			92.7	68.2	6.0
	41 – 80	87.0	77.4	5.7	----------			76.5	70.8	4.9	----------		
	81 – 120	76.1	66.4	4.9	75.2	66.7	4.9	----------			----------		
# 4	1 – 40	87.8	56.0	5.7	----------			----------			96.7	58.4	6.3
	41 – 80	70.1	54.2	4.6	70.9	54.2	4.6	----------			----------		
	81 – 120	65.3	63.1	4.2	----------			71.3	66.9	4.6	----------		
# 5	1 – 40	79.1	55.3	5.1	89.8	59.6	5.8	----------			----------		
	41 – 80	75.2	54.1	4.9	----------			----------			82.5	58.1	5.4

To analyze the results from the previous table, some statistics were performed namely the ANOVA. For each parameter, SV, EF and CO, the ANOVA table was created evaluating if the considered image view could affect the parameter quantification and, consequently the final diagnosis. The parameters values used are the ones obtained from U-Net_FilledMasks_50000.

ANOVA Table – Stroke Volume

Source	SS	df	MS	F	Prob>F
Groups	151.26	2	75.63	0.17	0.8484
Error	4984.83	11	453.167		
Total	5136.09	13			

ANOVA Table – Ejection Fraction

Source	SS	df	MS	F	Prob>F
Groups	266.72	2	133.36	0.84	0.4561
Error	1738.45	11	158.041		
Total	2005.17	13			

ANOVA Table – Cardiac Output

Source	SS	df	MS	F	Prob>F
Groups	0.6606	2	0.33032	0.17	0.8463
Error	21.4515	11	1.95014		
Total	22.1121	13			

Figure 4.25 ANOVA tables for the parameter SV, EF and CO. The table entry "Prob>F" returns the p-value, considering a 5% significance level (α=0.05).

Additional paired t-tests were performed to evaluate the influence of the used network on the final SV, EF and CO values.

Table 4.9 Paired t-tests results for each parameter, SV, EF, and CO, comparing each pair of networks: U-Net_FilledMasks_50000 vs U-Net_2CH, U-Net_FilledMasks_50000 vs U-Net_3CH and U-Net_FilledMasks_50000 vs U-Net_4CH.

	Paired t-test – P-value ($\alpha = 0.05$)		
	U-Net_FilledMasks_50000 *vs* U-Net_2CH	**U-Net_FilledMasks_50000 *vs* U-Net_3CH**	**U-Net_FilledMasks_50000 *vs* U-Net_4CH**
SV	0.945	0.946	0.416
EF	0.947	0.967	0.606
CO	0.939	0.962	0.429

4.4 Regression Data Simulation

The simulated data used during the regression network training, validation and testing is presented next. The AIF was generated using a gamma function, the IRF came from equation 2.13 after giving a range of possible values for K^{trans} and establishing v_e, and $C_{myocardium}$ was the result of the convolution between these last two curves (equation 2.11).

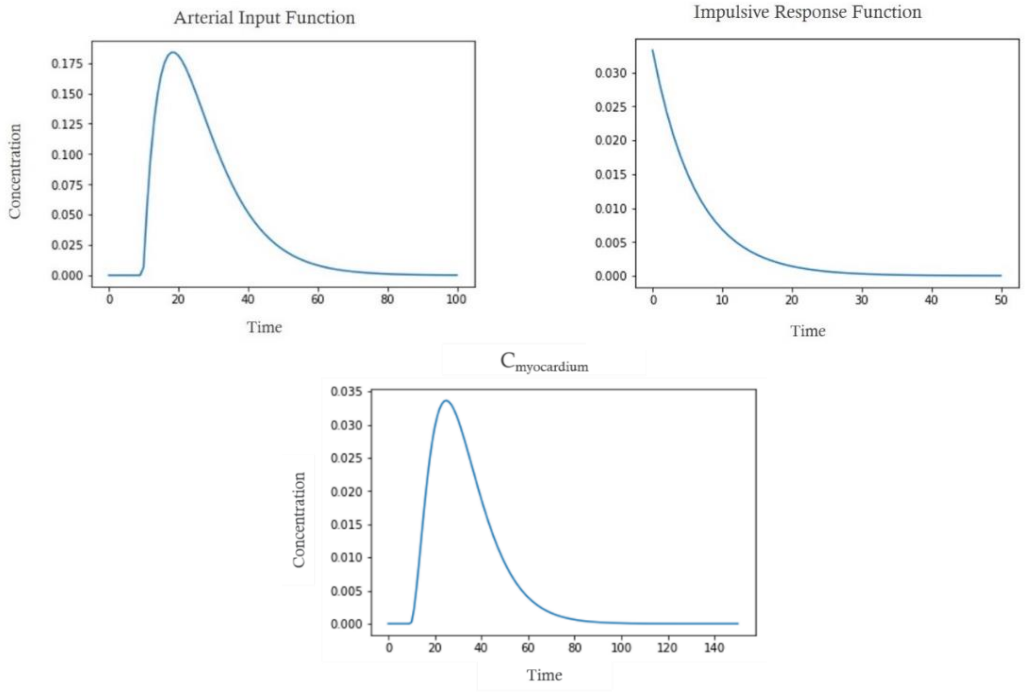


Figure 4.26 Generated regression data: AIF, IRF and $C_{myocardium}$. These generated curves were used to create the training, validation and test sets used during the regression scenario.

4.5 K^{trans} Obtained from Regression

The regression network was trained following the configuration in Table 3.8, obtaining the following loss curves. Here are also presented two network predictions for different validation AIF and $C_{myocardium}$ curves.

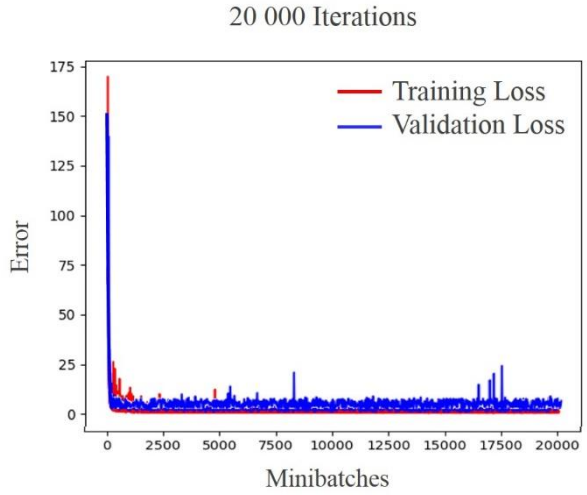


Figure 4.27 Training (red curve) and validation (blue curve) errors for the regression network. This network was only trained for 20 000 iterations.

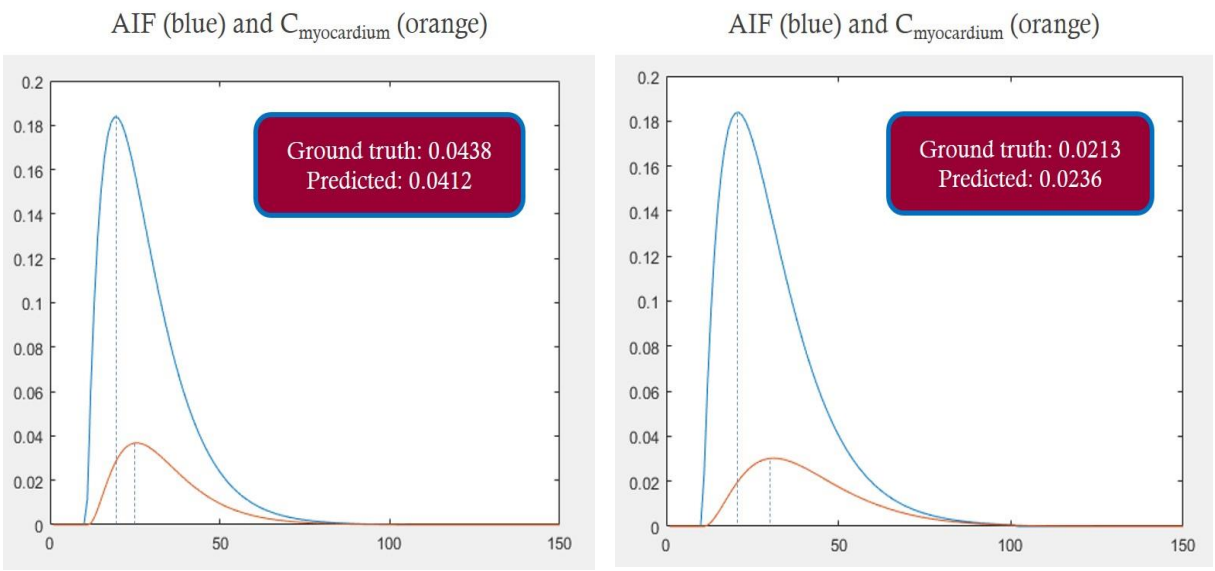


Figure 4.28 K^{trans} ground truth values and predictions obtained from the regression network. Each graph represents one validation example, the blue curve is the AIF and the orange is $C_{myocardium}$, used for the prediction.

The RMSE was calculated (Table 4.10), accordingly to equation 2.10, together with the R^2 from the fitted line to the validation data (300 observations), in order to evaluate how good the model was at making predictions.

Table 4.10 RMSE and R^2 calculated from the model fitted to the 300 validation observations.

RMSE	R^2
0.0075	0.5561

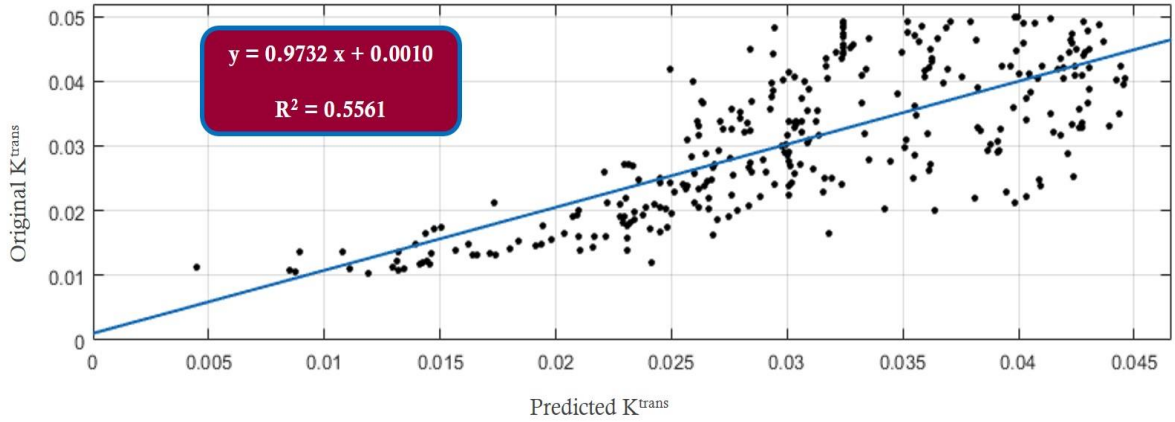


Figure 4.29 300 validation observations for the K^{trans} and respective fitted model. The black dots represent the predicted K^{trans} comparing to the blue fitting line.

The IRF temporal behavior reflects the myocardial condition of the patient. Figure 4.30 shows the behavior of 4 different IRFs: 2 of them correspond to the previous predicted examples (red and blue curves) and the other two represent limit situations where the patient is either in good myocardial perfusion condition (green curve) or not, showing some degree of occluded coronary arteries (purple curve).

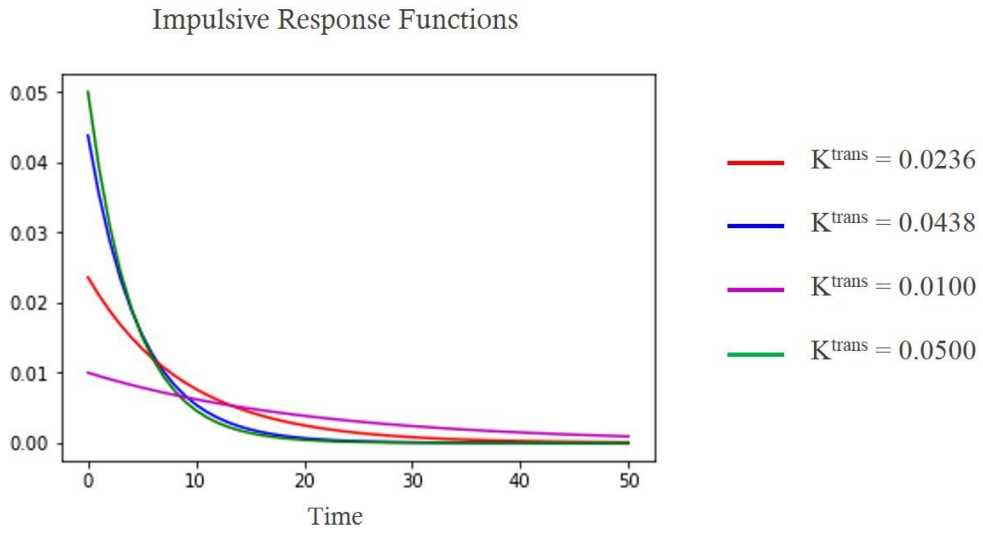


Figure 4.30 IRFs temporal behavior. Depending on the K^{trans}, the IRF will behave in a different way, from which is possible to evaluate the patient's myocardial perfusion condition.

5 Discussion

Starting with the CMR data, in spite of all the advices and tips gotten from a specialist in cardiac images analysis, the contours were not drawn by a clinical professional. Therefore, all the results that depend on the initial contours, such as LV areas, distance maps and 3 points labels or the LV's center of mass, might not be very precise due to contour imperfections. However, in further studies, it is possible to make changes on these structures, leading to more accurate segmentations and probably slightly improved results.

A. U-Net_20000 and U-Net_50000

Regarding networks U-Net_20000 and U-Net_50000 training, which was performed using the original training set, that contained the three types of views (2 CH-LA, 3 CH-LA and 4 CH-LA), and used distance maps as labels to predict the LV contour, the obtained graphics for the training and validation errors were very similar. Even though the maximum value for both training and validation errors from U-Net_20000 being greater than the errors from U-Net_50000, in both situations the errors approach zero quickly and keep this low during the whole training session. Another consideration to be made is that the relationship between the two errors curves follows the 2 presented properties: (1) the training error must be as low as possible and (2) the gap between the training and validation errors should be as small as possible, avoiding both under and overfitting.

Analyzing U-Net_20000 and U-Net_50000 LV contour predictions, the one from the latter network adapts better to the LV, as it is shown by the red arrow in Figure 4.7 (the contour is more curved in this area that the one predicted by U-Net_20000). Also, the U-Net_50000 predicted contour looks smoother than the other in some areas.

The Dice Coefficient was measured to quantify how good the predictions were. It is more complicated to measure the Dice when the area to segment is smaller, i.e. it is more difficult to predict a single point on the image than a circular area, for example. As mentioned, the closer this score gets to 1, the better, meaning that the prediction is more similar to the ground truth. For both networks, regarding the Dice score evolution during training and validation it changes from 0 to its mean value very fast and stays around this value during the rest of training session, meaning that the network is learning quickly, regarding the validation Dice score this is slightly smaller than the training one since the networks are making predictions on images never seen before and updating the hyperparameters at the same time. The U-Net_50000's validation predictions Dice score is slightly smaller than the one from U-Net_20000's prediction, meaning that there is less overlap between the prediction and the ground truth. This can be explained by the fact that the U-Net_50000's predicted contour looks thinner than the ground truth, showing a better fitting to the LV.

The training and validation errors and the Dice Coefficient from both networks suggest that their performance is quite good and very similar.

B. U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000

Considering now the U-Net_FilledMasks_20000 and U-Net_FilledMasks_50000 training, a similar analysis was performed. These networks were trained using the original training set, used LV areas as labels and looking at the errors' graphs, the "Number of Iterations" influence on the training session can

be understood: the more iterations the training considers, the more time the network has to reach an optimal error, i.e. the lowest error as possible but keeping track of the 2 conditions to avoid under and overfitting. The network trained over 50 000 iterations has a better commitment between these two requirements since the validation error from the network trained over 20 000 iterations is higher and fluctuates a lot in the beginning of the training session, supporting the fact that with more iterations the errors have more time to improve and stabilize since the network learns during a longer period.

From Figure 4.11 it is possible to see that the U-Net_FilledMasks_50000's LV area prediction adapts better to the LV, as indicated by the red arrows. This fact can be confirmed by the mean validation Dice Coefficient which is higher for the U-Net_FilledMasks_50000's prediction, indicating that this prediction overlaps best with the ground truth label. Still considering these mean Dice scores and comparing them with the ones from the U-Net_20000 and U-Net_50000 networks, these ones get closer to 1, supporting the fact that it is more difficult to predict a smaller structure, like the LV contour, than a larger one, like the LV area.

The evolution of the training and validation Dice Coefficients from both networks suggest that the U-Net_FilledMasks_50000's performance is better than the U-Net_FilledMasks_20000's one. Therefore, the first was the selected network to use in the comparison results analyzed further ahead.

C. U-Net_2CH, U-Net_3CH and U-Net_4CH

The networks U-Net_2CH, U-Net_3CH and U-Net_4CH were trained using training sets different from the original one. Respectively, they used training sets only containing 2 CH-LA, 3 CH-LA and 4 CH-LA images and all of them predicted LV areas. The three errors' graphs were very similar among them, therefore only one was presented since the conclusions are the same. The validation error does not reach a level very close to zero during the 50 000 iterations. This evidence, attached to the fact that the used training sets were smaller and did not have as much variation as the original training set, may suggest that the U-Net_FilledMasks_50000 is better at predicting the LV area.

Looking at each network predictions (Figure 4.15), independently of the considered view, the predicted areas seem to present more blurred outlines than the U-Net_FilledMasks_50000's prediction. The mean Dice scores for validation predictions from U-Net_2CH and U-Net_4CH are better than the mean Dice for U-Net_3CH, what could be explained by the fact that the U-Net_3CH training set contains less images, and consequently less variability. The Dice score for U-Net_2CH is slightly greater than the one for the U-Net_FilledMasks_50000 network, suggesting that the first network might be better at predicting the LV areas from 2 CH-LA images. The other Dice scores are smaller than the one for U-Net_FilledMasks_50000 indicating a less perfect overlap with the ground truth label, leading to a conclusion that might favor this latter network instead of U-Net_3CH and U-Net_4CH.

D. Three points U-Net

The 8th network was trained using the original training set and predicted three important points from the LV: the apex and 2 valve points. The Dice scores temporal evolution for each class confirm that this coefficient initial value rapidly changes its value from zero to values around the mean Dice score and it stays around it during the rest of the training session, revealing that the network is actually learning.

The network prediction (Figure 4.17) shows the obtained blobs and the final points' coordinates after submitting the prediction to the created script action. Comparing the point's coordinates from the

ground truth with the predicted ones, the coincidence between the first valve point (159, 208; 160, 208) is the best of the three, with the one between the apex point being less good. For the validation predictions the mean Dice Coefficient for each point, i.e. each class, was calculated with all scores being much smaller than 1, confirming the already stated Dice property that is more complicated to predict points since they are smaller and more specific structures. However, in this context, the highest Dice score corresponds to first valve point and the lowest to the apex point confirming the aforementioned statement about the point's coordinates values and suggesting that the apex point prediction might be more difficult to achieve. When this 8th network was being trained, it received the images' labels following a specific order which was [Apex Point → Second Valve Point → First Valve Point] and maybe for this reason the Dice score for the first valve point is the highest, since the network started by predicting the apex and second valve points and used that information to predict, in the end, the first valve point.

The results coming from this network are used to measure the LV long-axis, so it is expected that the points' final coordinates will influence this variable value.

E. Ventricular volume – Time curves and Functional Parameters

All the presented results regarding the Ventricular volume – Time curves and the functional parameters quantification were obtained using the test set, which has no labels. These are the results that the clinical experts should evaluate.

From the Ventricular volume – Time curves, which were obtained after calculating the LV volume from the LV area predictions for each phase of each cardiac cycle, the 6 curves' behavior strongly corresponds to the expected (Figure 2.1). For each of the 3 pairs of curves (U-Net_2CH and U-Net_FilledMasks_50000, U-Net_3CH and U-Net_FilledMasks_50000, U-Net_4CH and U-Net_FilledMasks_50000), the behavior is also very similar among each. Some curves have points that might be slightly dislocated from the expected curve shape (Figure 4.24 – blue curve, phase 23) and that might be due to the LV area or the three points predictions. Since the ventricular volumes suffer the influence of these two predictions, it's value and all further conclusions are biased. For this reason and to help deciding which of the three previous networks' pairs provided the best LV area predictions, paired t-tests were performed based on the measured LV areas and not volumes.

For these three paired t-tests the null hypothesis was "There are no significant differences in the mean of the predicted LV area values for a given cardiac cycle" and since all the three p-values were greater than the 5% significance level, it means that it does not matter if the LV area prediction comes from the U-Net_FilledMasks_50000 or the U-Net_2CH/ U-Net_3CH/ U-Net_4CH, since there is no difference in the mean of the LV area values for a determined cardiac cycle. This result does not deny that U-Net_FilledMasks_50000 might be a slightly better network than U-Net_2CH/ U-Net_3CH/ U-Net_4CH (part C), but it gives credit to these three view-specific networks allowing to use any of the four in future projects. This conclusion is also supported analyzing the mean Dice scores for these 4 networks which are all close.

To evaluate the parameters' values from the test set, reference intervals were created from the original training data using the same method applied to the test set. These intervals were created for 3 variables: SV, EF and CO, and ranged from the (Mean Reference Value – SD) to (Mean Reference Value + SD).

With the SV, EF and CO quantified from the test set and using LV area predictions from the U-Net_FilledMasks_50000, U-Net_2CH, U-Net_3CH and U-Net_4CH networks, the majority of the values are within the reference intervals created from the training set. To analyze these results (Table 4.8) paired t-tests and ANOVAs were performed. The ANOVA tests were performed to check the image view influence on each parameter's values, under the null hypothesis of "There are no significant differences in the mean of the obtained SV/ EF/ CO between the three different groups (2 CH-LA, 3 CH-LA, 4 CH-LA) of the independent variable "image view"". Looking at the p-values, the 3 are greater than the 5% significance level, meaning that it is not important which view is being used to quantify the parameters. It makes sense since the parameter value should be the same, for the same patient, regardless of the cardiac cycle images view.

Paired t-tests were also performed to evaluate which of the 4 networks' predictions could be used to provide the best parameters' values, i.e. a paired t-test was performed for each parameter to check if the final values were better when obtained from the U-Net_FilledMasks_50000 or from one of the view specific networks. The obtained p-values state that the parameters' values obtained from U-Net_FilledMasks_50000 are not better than the ones obtained from the view specific networks, a result that coincides with the result from the other paired t-test mentioned above, where was established that the LV area prediction provenience is irrelevant.

F. Regression

The generated perfusion data meets the expectations: the AIF has a higher maximum and its peak happens sooner than $C_{myocardium}$ and the IRF shows what happens in the myocardium during time, which is the contrast agent elimination.

The regression network was trained on a generated data set containing AIF and $C_{myocardium}$ curves and predicted the volume transfer constant, K^{trans}, given the EES fraction, v_e. In spite of the slightly high validation error, which could be lowered by training the network during more iterations, the two conditions describing the relationship between the two errors curves are met, avoiding under and overfitting.

The K^{trans} predictions only fail on the thousandths order (Figure 4.28), showing that the regression network has some predictive skill. This slight failure might be due to the training and the number of iterations, however the predictions are not very far from the real number. For the 300 K^{trans} predictions the R^2 indicates that the fitted model explains around 56% of the variance meaning that more than half of the predictions coincide with the original values. These results confirm that the regression network has a predictive power that can still be improved.

With the K^{trans} the IRF can be plotted to assess the coronaries occlusion level. Some IRFs were plotted and the difference is evident in these functions' behavior: the higher the K^{trans} is, the lower the coronary occlusion level and the faster the myocardial tissue will eliminate the contrast agent. In a limit situation, K^{trans} would be zero and the IRF plot would be a horizontal line.

6 Conclusions

The goal of this project was to investigate the potential of DL algorithms to analyze LA CMR images through segmentation, registration and quantification of some functional parameters such as SV, EF and CO.

Fully automatic segmentation is still a challenge since it always requires some minor corrections in the final segmentation. During this project 2 CNNs were trained to fully automatically segment the LV contour and since it is less difficult and error-prone to segment the LV area, 4 CNNs were trained and can be used in future projects since they present results for the LV segmentation that can be leveled with state of the art [60] – [61] ones. It is worth noticing that none of these state of the art references use LA images and both of them rely on a previous ROI establishment before feeding the image to the network. Comparing to the recent results from [62] where 2 CH-LA and 4 CH-LA images were used together with SA ones, this project's results are also leveled with them.

To the best of my knowledge, the existence of a network that predicts the three most important landmarks for segmentation and parameters' quantification is not known. Here, a CNN is trained to provide predictions for these points, achieving quite good results, facilitating the creation of a user independent framework and opening space to train more view specific networks to predict these three points.

The obtained values for the functional parameters do not depend on the LV area prediction provenience. This fact favors the creation of a user independent framework that directly quantifies the SV, EF and CO from the input LA image. These results, together with more medical information about the patient, facilitate the clinician final judgement about the diagnosis.

Regarding the regression results of this project these are promising, however there is way more work to do by starting to collect real perfusion labeled data, with the most variability as possible, and train more regression networks during more time in order to better assess the myocardial condition of the patient.

In conclusion, the initial project's goal was achieved and some more work can be left to do in the future such as the correction of the LV contours used as labels, the creation of new label sets that include the ground truth value for the LV volume, alterations in the U-Net architecture by adding or removing more layers and use these networks to make predictions on SA images.

7 References

[1] "WHO | Cardiovascular diseases (CVDs)," *WHO*. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs317/en/. [Accessed: 01-Nov-2017].

[2] H. Futamatsu *et al.*, "Usefulness of cardiac magnetic resonance imaging for coronary artery disease detection," *Minerva Cardioangiol.*, vol. 55, no. 1, pp. 105–114, Feb. 2007.

[3] M. Jerosch-Herold, "Quantification of myocardial perfusion by cardiovascular magnetic resonance," *J. Cardiovasc. Magn. Reson.*, vol. 12, p. 57, 2010.

[4] M. Breeuwer, "Quantification of atherosclerotic heart disease with cardiac MRI," *MedicaMundi*, vol. 49, no. 2, pp. 30–38, Aug. 2005.

[5] M. Breeuwer, G. Hautvast, S. Higgins, and E. Nagel, "Simplifying cardiac MR analysis," *MedicaMundi*, vol. 52, no. 2, pp. 68–76, 2008.

[6] D. Sidebotham and I. J. Le Grice, "Chapter 1 - Physiology and Pathophysiology," in *Cardiothoracic Critical Care*, Philadelphia: Butterworth-Heinemann, 2007, pp. 3–27.

[7] "The Concept | Hemologic." [Online]. Available: http://www.hemologic.com/the-concept. [Accessed: 25-Apr-2018].

[8] "OsiriX: Ejection Fraction." [Online]. Available: http://www.osirix-viewer.com/Documentation/Guides/Plugins/EjectionFraction/index.html. [Accessed: 24-Apr-2018].

[9] C. Petitjean and J.-N. Dacher, "A review of segmentation methods in short axis cardiac MR images," *Med. Image Anal.*, no. 15, pp. 169–184, 2011.

[10] D. Grosgeorge, C. Petitjean, J. Caudron, J. Fares, and J.-N. Dacher, "Automatic cardiac ventricle segmentation in MR images: a validation study," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 6, no. 5, pp. 573–581, Sep. 2011.

[11] S. P. O'Brien, O. Ghita, and P. F. Whelan, "A Novel Model-Based 3D +Time Left Ventricular Segmentation Technique," *IEEE Trans. Med. Imaging*, vol. 30, no. 2, pp. 461–474, Feb. 2011.

[12] T. Hazirolan *et al.*, "Comparison of short and long axis methods in cardiac MR imaging and echocardiography for left ventricular function," *Diagn. Interv. Radiol.*, pp. 33–38, 2007.

[13] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.

[14] D. Sullivan, "How Machine Learning Works, As Explained By Google," 2015. [Online]. Available: https://martechtoday.com/how-machine-learning-works-150366.

[15] P. Domingos, "A Few Useful Things to Know about Machine Learning," *Mag. Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[16] A. Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/classification/. [Accessed: 20-Nov-2017].

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[19] N. de Freitas, "Machine Learning." [Online]. Available: https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/. [Accessed: 15-Dec-2017].

[20] A. Bronshtein, "Train/Test Split and Cross Validation in Python," *Towards Data Science*, 17-May-2017. [Online]. Available: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6. [Accessed: 29-Nov-2018].

[21] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *ArXiv171005941 Cs*, Oct. 2017.

[22] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Stat. Plan. Inference*, vol. 90, no. 2, pp. 227–244, Oct. 2000.

[23] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ArXiv150203167 Cs*, Feb. 2015.

[24] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, Jan. 1988.
[25] S. Lobregt and M. A. Viergever, “A discrete dynamic contour model,” *IEEE Trans. Med. Imaging*, vol. 14, no. 1, pp. 12–24, 1995.
[26] G. Hautvast, S. Lobregt, M. Breeuwer, and F. Gerritsen, “Automatic Contour Propagation in Cine Cardiac Magnetic Resonance Images,” *IEEE Trans. Med. Imaging*, vol. 25, no. 11, pp. 1472–1482, Nov. 2006.
[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, USA, 2012, pp. 1097–1105.
[29] I. J. Goodfellow *et al.*, “Generative Adversarial Networks,” *ArXiv14062661 Cs Stat*, Jun. 2014.
[30] M. J. M. Chuquicusma, S. Hussein, J. Burt, and U. Bagci, “How to Fool Radiologists with Generative Adversarial Networks? A Visual Turing Test for Lung Cancer Diagnosis,” *ArXiv171009762 Cs Q-Bio*, Oct. 2017.
[31] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ArXiv14091556 Cs*, Sep. 2014.
[32] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *ArXiv14094842 Cs*, Sep. 2014.
[33] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *ArXiv14114038 Cs*, Nov. 2014.
[34] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *ArXiv150504597 Cs*, May 2015.
[35] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation,” *ArXiv160606650 Cs*, Jun. 2016.
[36] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation,” *ArXiv160604797 Cs*, Jun. 2016.
[37] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. de Vries, M. J. N. L. Benders, and I. Išgum, “Automatic Segmentation of MR Brain Images With a Convolutional Neural Network,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1252–1261, May 2016.
[38] K. Kamnitsas *et al.*, “Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation,” *Med. Image Anal.*, vol. 36, pp. 61–78, Feb. 2017.
[39] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, p. 115, Feb. 2017.
[40] V. Gulshan *et al.*, “Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs,” *JAMA*, vol. 316, no. 22, pp. 2402–2410, Dec. 2016.
[41] H. Salehinejad, J. Baarbe, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” *ArXiv180101078 Cs*, Dec. 2017.
[42] J. M. Wolterink, T. Leiner, B. D. de Vos, R. W. van Hamersvelt, M. A. Viergever, and I. Išgum, “Automatic coronary artery calcium scoring in cardiac CT angiography using paired convolutional neural networks,” *Med. Image Anal.*, vol. 34, pp. 123–136, Dec. 2016.
[43] R. P. K. Poudel, P. Lamata, and G. Montana, “Recurrent Fully Convolutional Neural Networks for Multi-slice MRI Cardiac Segmentation,” *ArXiv160803974 Cs Stat*, Aug. 2016.
[44] J. Caudron, J. Fares, F. Bauer, and J.-N. Dacher, “Evaluation of Left Ventricular Diastolic Function with Cardiac MR Imaging,” *RadioGraphics*, vol. 31, no. 1, pp. 239–259, Jan. 2011.
[45] R. J. van der Geest, E. Jansen, V. G. M. Buller, and J. H. C. Reiber, “Automated detection of left ventricular epi- and endocardial contours in short-axis MR images,” in *Computers in Cardiology 1994*, 1994, pp. 33–36.
[46] T. O’Donnell, G. Funka-Lea, H. Tek, M.-P. Jolly, M. Rasch, and R. Setser, “Comprehensive Cardiovascular Image Analysis Using MR and CT at Siemens Corporate Research,” *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 165–178, Nov. 2006.
[47] T. Hazirolan *et al.*, “Comparison of short and long axis methods in cardiac MR imaging and echocardiography for left ventricular function,” *Diagn. Interv. Radiol. Ank. Turk.*, vol. 13, no. 1, pp. 33–38, Mar. 2007.

[48] X. Zhuang, K. S. Rhode, R. S. Razavi, D. J. Hawkes, and S. Ourselin, "A Registration-Based Propagation Framework for Automatic Whole Heart Segmentation of Cardiac MRI," *IEEE Trans. Med. Imaging*, vol. 29, no. 9, pp. 1612–1625, Sep. 2010.
[49] Y. Tsadok, Y. Petrank, S. Sarvari, T. Edvardsen, and D. Adam, "Automatic segmentation of cardiac MRI cines validated for long axis views," *Comput. Med. Imaging Graph.*, vol. 37, no. 7, pp. 500–511, Oct. 2013.
[50] G. L. T. F. Hautvast, M. Breeuwer, S. Lobregt, and F. A. Gerritsen, "Automatic exclusion of papillary muscles and trabeculae from blood volume measurements in cine cardiac magnetic resonance images," in *2006 Computers in Cardiology*, 2006, pp. 57–60.
[51] M. L. Chuang *et al.*, "Correlation of trabeculae and papillary muscles with clinical and cardiac characteristics and impact on CMR measures of LV anatomy and function," *JACC Cardiovasc. Imaging*, vol. 5, no. 11, pp. 1115–1123, Nov. 2012.
[52] "Arterys - Medical Imaging Cloud AI." [Online]. Available: https://arterys.com/. [Accessed: 10-Jan-2018].
[53] A. Mehrtash *et al.*, "Bolus arrival time and its effect on tissue characterization with dynamic contrast-enhanced magnetic resonance imaging," *J. Med. Imaging*, vol. 3, no. 1, pp. 14503–16, Jan. 2016.
[54] P. S. Tofts and A. G. Kermode, "Measurement of the blood-brain barrier permeability and leakage space using dynamic MR imaging. 1. Fundamental concepts," *Magn. Reson. Med.*, vol. 17, no. 2, pp. 357–367, Feb. 1991.
[55] "U-Net: Convolutional Networks for Biomedical Image Segmentation." [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/. [Accessed: 30-Aug-2018].
[56] The Theano Development Team *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," *ArXiv160502688 Cs*, May 2016.
[57] "Welcome to Lasagne — Lasagne 0.2.dev1 documentation." [Online]. Available: https://lasagne.readthedocs.io/en/latest/#. [Accessed: 30-Aug-2018].
[58] K. H. Zou *et al.*, "Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index," *Acad. Radiol.*, vol. 11, no. 2, pp. 178–189, Feb. 2004.
[59] P. S. Tofts, "T1-weighted DCE imaging concepts: Modelling, acquisition and analysis," *MAGNETOM Flash*, vol. 3, pp. 30–39, Jan. 2010.
[60] M. Nasr-Esfahani *et al.*, "Left Ventricle Segmentation in Cardiac MR Images Using Fully Convolutional Network," *ArXiv180207778 Cs*, Feb. 2018.
[61] X. Yang, Z. Zeng, and S. Yi, "Deep convolutional neural networks for automatic segmentation of left ventricle cavity from cardiac magnetic resonance images," *IET Comput. Vis.*, vol. 11, pp. 643–649, Jun. 2017.
[62] G. Luo, S. Dong, K. Wang, W. Zuo, S. Cao, and H. Zhang, "Multi-Views Fusion CNN for Left Ventricular Volumes Estimation on Cardiac MR Images," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 9, pp. 1924–1934, Sep. 2018.

8 Appendix A

The following set of images show the LV segmentations obtained from U-Net_FilledMasks_50000 for one of the 14 cardiac cycles from the test set.